PATENT **MS150905.01/MSFTP166US**

CERTIFICATE OF MAILING

I hereby certify that this correspondence (along with any paper referred to as being attached or enclosed) is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-14501.

3-21-05
Date:

Himanshu S. Amin

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:

Applicant(s): Susan T. Dumais, *et al.*

Serial No: 09/893,827

Filing Date: June 28, 2001

Examiner: Brian D. Goddard

Art Unit: 2171

Title: SYSTEM, REPRESENTATION, AND METHOD PROVIDING MULTILEVEL INFORMATION RETRIEVAL WITH CLARIFICATION DIALOG

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION UNDER 37 C.F.R. §1.131

Dear Sir:

The undersigned, the inventors of the claimed invention of the subject patent application, declare and say as follows:

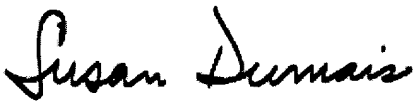
We are the inventors of the claims of the above-identified patent application. This Declaration is submitted to establish reduction to practice of the invention described and claimed in the above-captioned application in the United States at a date prior to May 15, 2000, which is the effective date of Johnson, *et al.* (U.S. Patent 6,567,805 B1).

To establish reduction to practice of the invention claimed in the above-identified application prior to May 15, 2000, a copy of two disparate papers created prior to the effective date of Johnson, *et al*, are enclosed as exhibits A and C, respectively. Exhibit A describes a hierarchical information classification and information retrieval system, and describes building a top-level classifier and a sublevel classifier on pages 3-7. Exhibit C describes on page 4 how probabilities that a page belongs within a certain category are utilized as a parameter for presenting information to a user, and further illustrates receiving queries on page 1. Exhibits A and C sufficiently disclose the invention as recited in independent claims 1, 27, and 39. Moreover, exhibits A and C include diagrams and screenshots that clearly illustrate that the claimed invention was reduced to practice prior to the effective date of Johnson, *et al.* In particular, exhibits A and C include tables, graphs, and screenshots that clearly delineate that the claimed invention was reduced to practice and tested prior to May 15, 2000. Screenshots displaying dates of creation associated with exhibits A and C are also enclosed as exhibits B and D, respectively. Specifically, exhibit B illustrates that content of exhibit A was created no later than April 29, 2000, and exhibit D illustrates that content of exhibit C was created no later than April 7, 2000.

For further evidence of reduction to practice prior to the effective date of Johnson, *et al.,* slideshow presentations created prior to such date are enclosed as exhibits E and G, respectively, and screenshots illustrating dates of creation are enclosed herein as exhibits F and H, respectively. Exhibit E relates to building classifiers for utilization in automatically classifying information in a hierarchical manner, and exhibit G illustrates searching based upon a query and hierarchically structuring topics relating to the query. Exhibits E and G further include screenshots and results associated with one or more aspects of the claimed invention. Also enclosed is a copy of relevant portions of a disclosure meeting summary document dated October 10, 2000, labeled exhibit I. Certain information, such as proprietary information and other non-relevant information has been removed from these exhibits.

We hereby declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Respectfully submitted,



Mar 16, 2005

Susan T. Dumais Date

March 16, 2005

Eric J. Horvitz Date

EXHIBIT A

Hierarchical Classification of Web Content

Susan Dumais
Microsoft Research
One Microsoft Way
Redmond, WA 99802 USA
sdumais@microsoft.com

Hao Chen
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720-1776 USA
hchen@cs.berkeley.edu

ABSTRACT

This paper explores the use of hierarchical structure for classifying a large, heterogeneous collection of web content. The hierarchical structure is initially used to train different second-level classifiers. In the hierarchical case, a model is learned to distinguish a second-level category from other categories within the same top level. In the flat non-hierarchical case, a model distinguishes a second-level category from all other second-level categories. Scoring rules can further take advantage of the hierarchy by considering only second-level categories that exceed a threshold at the top level.

We use support vector machine (SVM) classifiers, which have been shown to be efficient and effective for classification, but not previously explored in the context of hierarchical classification. We found small advantages in accuracy for hierarchical models over flat models. For the hierarchical approach, we found the same accuracy using a sequential Boolean decision rule and a multiplicative decision rule. Since the sequential approach is much more efficient, requiring only 14%-16% of the comparisons used in the other approaches, we find it to be a good choice for classifying text into large hierarchical structures.

KEYWORDS

Text classification, text categorization, classification, support vector machines, machine learning, hierarchical models, web hierarchies

INTRODUCTION

With the exponential growth of information on the internet and intranets, it is becoming increasingly difficult to find and organize relevant materials. More and more, simple text retrieval systems are being supplemented with structured organizations. Since the 19th century, librarians have used classification systems like Dewey and Library of Congress subject headings to organize vast amounts of information. More recently, web directories such as Yahoo! and LookSmart have been used to classify web pages. Structured directories support browsing and search, but the manual nature of the directory compiling process makes it difficult to keep pace with the ever increasing amount of information. Our work looks at the use of automatic classification methods to supplement human effort in creating structured knowledge hierarchies.

A wide range of statistical and machine learning techniques have been applied to text categorization, including multivariate regression models [8,22], nearest neighbor classifiers [26], probabilistic Bayesian models [13, 16], decision trees [16], neural networks [22, 25], symbolic rule learning [1, 4], and support vector machines [7,12]. These approaches all depend on having some initial labeled training data from which category models are learned. Once category models are trained, new items can be added with little or no additional human effort.

Although many real world classification systems have complex hierarchical structure (e.g., MeSH, U.S. Patents, Yahoo!, LookSmart), few learning methods capitalize on this structure. Most of the approaches mentioned above ignore hierarchical structure and treat each category or class separately, thus in effect "flattening" the class structure. A separate binary classifier is learned to distinguish each class from all other classes. The binary classifiers can be considered independently, so an item may fall into none, one, or more than one category. Or they can be considered as an m-ary problem, where the best matching category is chosen. Such simple approaches work rather well on small problems, but they are likely to be difficult to train when there are a large number of classes and a very large number of features. By utilizing known hierarchical structure, the classification problem can be decomposed into a set of smaller problems corresponding to hierarchical splits in the tree. Roughly speaking, one first learns to distinguish among classes at the top level, then lower level distinctions are learned only within the appropriate top level of the tree. Each of these sub-problems can be solved much more efficiently, and hopefully more accurately as well.

The use of a hierarchical decomposition of a classification problem allows for efficiencies in both learning and representation. Each sub-problem is smaller than the original problem, and it is sometimes possible to use a much smaller set of features for each [13]. The hierarchical structure can also be used to set the negative set for discriminative training and at classification time to combine information from different levels. In addition, there is some

evidence that decomposing the problem can lead to more accurate specialized classifiers. Intuitively, many potentially good features are not useful discriminators in non-hierarchical representations. Imagine a hierarchy with two top-level categories ("Computers" and "Sports"), and three subcategories within each ("Computers/Hardware", "Computers/Software", "Computers/Chat", "Sports/Chat", "Sports/Soccer", Sports/Football"). In a non-hierarchical model, a word like "computer" is not very discriminating since it is associated with items in "Computers/Software", "Computers/Hardware", and "Computers/Chat". In a hierarchical model, the word "computer" would be very discriminating at the first level. At the second level more specialized words could be used as features within the top-level "Computer" category. And, the same features could be used at the second level for two different top-level classed (e.g., "chat" might be a useful feature for both the category "Sports/Chat", and "Computers/Chat"). Informal failure analyses of classification errors for non-hierarchical models support this intuition. Many of the classification errors are for related categories (e.g., a page about "Sports/Soccer" might be confused with "Sports/Football", thus category specific features should improve accuracy).

Recently several researchers have investigated the use of hierarchies for text classification, with promising results. Our work differs from earlier work in a couple of important respects. First, we test the approach on a large collection of very heterogeneous web content, which we believe is increasingly characteristic of information organization problems. Second, we use a learning model, support vector machine (SVM), that has not previously been explored in the context of hierarchical classification. SVMs have been found to be more accurate for text classification than popular approaches like naïve Bayes, neural nets, and Rocchio [7, 12, 27]. We use a reduced-dimension binary-feature version of the SVM model that is very efficient for both initial learning and real-time classification, thus making it applicable to large dynamic collections. We will briefly review the earlier work and contrast it with ours.

RELATED WORK

Reuters

Much of the previous work on hierarchical methods for text classification uses the Reuters-22173 or Reuters-21578 articles. This is a rather small and tidy collection, and this alone is problematic for understanding how the approaches generalize to larger more complex internet applications. In addition, the Reuters articles are organized into 135 topical categories with no hierarchical structure. To study hierarchical models, researchers have added one level of hierarchical structure manually. Kohler and Sahami [13] generated a small hierarchical subset of Reuters-22173 by identifying labels that tended to subsume other labels (e.g., corn and wheat are subsumed by grain). The largest of their hierarchies consisted of 939 documents organized into 3 top-level and 6 second-level categories. They compared naïve Bayes, and two limited dependency Bayes net classifiers on flat and hierarchical models. Test documents were classified into the hierarchy by first filtering them through the single best matching first level class and then sending them to the appropriate second level. Note that errors made at the first level are not recoverable, so the system has to make k correct classification for a k-level hierarchy. They found advantages for the hierarchical models when a very small number of features (10) were used per class. For larger numbers of features (which will be required in many complex domains) no advantages for the hierarchical models were found.

Weigend et al. [25] also used the Reuters-22173 collection. An exploratory cluster analysis was first used to suggest an implicit hierarchical structure, and this was then verified by human assignments. They created 4 top-level categories (agriculture, energy, foreign exchange, and metals) and a 5th miscellaneous category that was not used for evaluation. The final evaluations were on 37 categories with at least 16 positive training examples. They used a probabilistic approach that frames the learning problem as one of function approximation for the posterior probability of the topic vector given the input vector. They used a neural net architecture and explored several input representations. Information from each level of the hierarchy is combined in a multiplicative fashion, so no hard decisions have to be made except at the leaf nodes. They found a 5% advantage in average precision for the hierarchical representation when using words, but not when using latent semantic indexing (LSI) features.

D'Alessio et al. [6] used the Reuters-21578 articles. The hierarchy they use comes from Hayes and Weinstein's original Reuters experiments [9]. It consists of 5 meta-category codes (economic indicator, currency, corporate, commodity, energy) that include all but three of the original categories. For their experiments they only considered articles that had a single category tag, and the 37 categories with more than 20 positive training examples. Their model requires hard assignment at each branch of the tree. The hierarchical model showed 2-4% improvements in precision and recall over the flat model, and modifications to the hierarchy led to advantages of 2-9%.

Ng et al. [19] also used a hierarchical version of Reuters that consisted of countries at the top level and two topical levels below that, but they did not compare their hierarchical model against a flat model. While all of this work is encouraging, the Reuters collection is small and very well organized compared with many realistic applications.

MeSH, U.S. patents

Some researchers have investigated text classification in domains that have rich hierarchical taxonomies (e.g., MeSH, IDC codes of diseases, U.S. patent codes). The size and complexity of the medical and patent hierarchies are like those used for web content. These hierarchies were

designed for controlled vocabulary tagging and they have been extensively refined over the years. Ruiz and Srinivasan [21] used a hierarchical mixture of experts to classify abstracts within the MeSH sub-category Heart. They learned classifiers at different levels of granularity, with the top-level distinctions serving as "gates" to the lower level "experts". They found small advantages for the hierarchical approach compared with a flat approach. Larkey [15] compared hierarchical and flat approaches for classifying patents in the Speech Signal Processing sub-category. She found no multilevel algorithms that performed significantly better than a flat one which chooses among all the speech classes.

Web

McCallum, et al. [17] describe some interesting experiments on three hierarchical collections -- Usenet news, the Science sub-category of Yahoo!, and company web pages. The Yahoo! sub-collection is most closely related to our experiments, although it is less diverse since all items come from the same top-level category. They used a probabilistic Bayesian framework (naïve Bayes), and a technique called "shrinkage" to improve parameter estimates for the probability of words given classes. The idea is to smooth the parameter estimate of a node by interpolation with all its parent nodes (given by the hierarchical organization of classes). This is especially useful for classes with small numbers of training examples. For the Usenet collection, the best performance and largest advantages of the hierarchical model over the flat model are observed for large numbers of features (10,000). For the Yahoo! sub-collection, accuracy is low and there is a much smaller difference between two approaches

Mladenic and Grobelnik [18] examined issues of feature selection for hierarchical classification of web content, but they did not compare hierarchical models with flat non-hierarchical models. Chakrabarti et al. [2] also experimented with web content as well as patent and Reuters data. They use hierarchical structure both to select features and to combine class information from multiple levels of the hierarchy. For the non-hierarchical case, they use a simple vector method with term weighting and no feature selection. They find no advantages for the hierarchical approach in Reuters, a small advantage in the patents collection, and did not test the difference for the Yahoo! sub-collection. But, because they used different representations, it is difficult to know whether differences are due to the number of features, feature selection, or the hierarchical classification algorithm.

Our work explores the use of hierarchies for classifying very heterogeneous web content. We use SVMs, which have been found to be efficient and effective for text classification, but not previously explored in the context of hierarchical classification. The efficiency of SVMs for both initial learning and real-time classification make them applicable to large dynamic collections like web content. We use the hierarchical structure for two purposes. First, we train second-level category models using different contrast sets (either within the same top-level category in the hierarchical case, or across all categories in the flat non-hierarchical case). Second, we combine scores from the top- and second-level models using different combination rules, some requiring a threshold to be exceeded at the top level before second level comparisons are made. The sequential approach is especially efficient for large problems, but how it compares in accuracy is not known. All of our models allow for each test item to be assigned to multiple categories.

WEB DATA SET AND APPLICATION

Application – Classifying web search results

We are interested in moving beyond today's ranked lists of results by organizing web search results into hierarchical categories. HCI researchers have shown usability advantages of structuring content hierarchically [3, 10, 14]. For information as varied and voluminous as the web it is necessary to create these structures automatically. Several groups have explored methods for *clustering* search results. Zamir and Etzioni [29] grouped web search results using suffix tree clustering, and Hearst and Pedersen [11] used Scatter/Gather to organize and browse search results. While these methods show some promise, the computational complexity of clustering algorithms limits the number of documents that can be processed, and generating names for the resulting categories is a challenge.

The basic idea behind our work is to use *classification* techniques to automatically organize search results into existing hierarchical structures. Classification models are learned offline using a training set of human-labeled documents and web categories. Classification offers two advantages compared to clustering – run time classification is very efficient, and manually generated category names are easily understood. To support our goal of automatically classifying web search results two constraints are important in understanding the classification problems we studied. First, we used just the short summaries returned from web search engines since it takes too long to retrieve the full text of pages in a networked environment. These automatically generated summaries are much shorter than the texts used in most classification experiments, and they are much noisier than other document surrogates like abstracts that some have worked with. Second, we focused on the top levels of the hierarchy since we believe that many search results can be usefully disambiguated at this level. We also developed an interface that tightly couples search results and category structure, and found large preference and performance advantages for automatically classified search results (see Chen and Dumais [3] for details). For the experiments reported in this paper, we focused on the top two levels of the hierarchy, and used short automatically generated summaries of web pages.

LookSmart Web Directory

For our experiments, we used a large heterogeneous collection of pages from LookSmart's web directory [30]. At the time we conducted our experiments in May 1999, the collection consisted of 370597 unique pages that had been manually classified into a hierarchy of categories by trained professional web editors. There were a total of 17173 categories organized into a 7-level hierarchy. We focused on the 13 top-level and 150 second-level categories.

Training/Test Selection

We selected a random 16% sample of the pages for our experiments. We picked this sampling proportion to ensure that even the smallest second-level categories would have some examples in our training set. We used 50078 pages for training, and 10024 for testing. Table 1 shows the number of pages in each top level category. Top level categories had between 578 and 11163 training examples, and second-level categories had between 3 and 3141 training examples. The wide range in number of pages in each category reflects the distribution in the LookSmart directory. Pages could be classified into more than one category. On average, pages were classified into 1.20 second-level categories and 1.07 first-level categories.

Category Name	Total	Train	Test
Automotive	4982	578	114
Business & Finance	31599	3508	703
Computers & Internet	46000	5718	1126
Entertainment & Media	88697	11163	2159
Health & Fitness	25380	3500	722
Hobbies & Interests	22959	3227	682
Home & Family	11484	1373	280
People & Chat	35157	3309	682
Reference & Education	58002	5574	1175
Shopping & Services	19667	2122	423
Society & Politics	38968	4855	946
Sports & Recreation	23559	3081	640
Travel & Vacations	43685	5409	1091
Total Unique	370597	50078	10024

Table 1- Training and Test Samples by Category

Pre-processing

A pre-processing module extracted plain text from each web page. In addition, the title, description and keywords fields from the META tag, and the ALT field from the IMG tag were also extracted if they existed because they provide useful descriptions of the web page. If a web page contained frames, text from each frame was extracted.

Since we were interested in classifying web search results, we needed to work with just short summary descriptions of web pages. We automatically generated summaries of each page and used this for evaluating our classification algorithms. Our summaries consisted of the title, the keywords, and either the description tag if it existed or the first 40 words of the body otherwise.

We did minimal additional processing on the text. We translated upper to lower case, used white space and punctuation to separate words, and used a small stop list to omit the most common words. No morphological or syntactic analyses were used.

After preprocessing, a binary vector was created for each page indicating whether each term appeared in the page summary or not. For each category 1000 terms were selected based on the mutual information between the term and category (details below). Unlike many text retrieval applications, term frequency and document length are not taken into account in this representation, because earlier experiments showed good performance with the binary representation for SVMs [7], and this representation improves efficiency.

TEXT CLASSIFICATION USING SVMs

Text classification involves a training phase and a testing phase. During the training phase, a large set of web pages with known category labels are used to train a classifier. An initial model is built using a subset of the labeled data, and a holdout set is used to identify optimal model parameters. During the testing or operational phase, the learned classifier is used to classify new web pages.

A support vector machine (SVM) algorithm was used as the classifier, because it has been shown in previous work to be both very fast and effective for text classification problems [7, 12, 27]. Vapnik introduced SVMs in his work on structural risk minimization [23, 24], and there has been a recent surge of interest in SVM classifiers in the learning community. In its simplest form, a linear SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum margin. The margin is the distance from the hyperplane to the nearest of the positive and negative examples.

Figure 1 shows an example of a simple two-dimensional problem that is linearly separable.

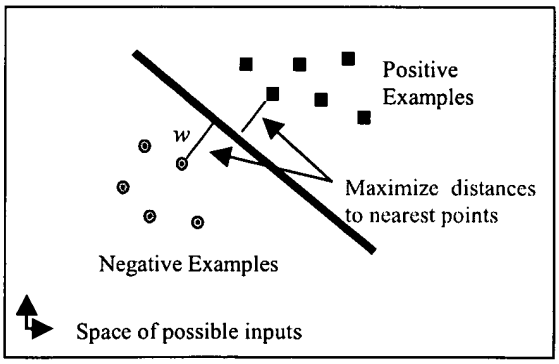


Figure 1 - Graphical Representation of a Linear SVM

In the linearly separable case maximizing the margin can be expressed as an optimization problem:

minimize $\frac{1}{2}\|\vec{w}\|^2$ subject to $y_i(\vec{w}\cdot\vec{x}_i - b) \geq 1, \forall i$

where x_i is the ith training example and y_i is the correct output of the SVM for the ith training example.

In cases where points are not linearly separable, slack variables are introduced that permit, but penalize, points that fall on the wrong side of the decision boundary. For problems that are not linearly separable, kernel methods can be used to transform the input space so that some non-linear problems can be learned. We used the simplest linear form of the SVM because it provided good classification accuracy, and is fast to learn and apply.

Platt [20] developed a very efficient method for learning the SVM weights. His sequential minimal optimization (SMO) algorithm breaks the large quadratic programming (QP) problem down into a series of small QP problems that can be solved analytically. Additional efficiencies can be realized because the training sets used for text classification are sparse and binary. Thus the SMO algorithm is nicely applicable for large feature and training sets.

Once the weights are learned, new items are classified by computing $\vec{w} \cdot \vec{x}$, where $\vec{w}$ is the vector of learned weights, and $\vec{x}$ is the input vector for a new document. With the binary representation we use, this amounts to taking the sum of the weights for features present in a document and this is very efficient.

After training the SVM, we fit a sigmoid to the output of the SVM using regularized maximum likelihood fitting, so that the SVM produces posterior probabilities that are directly comparable across categories.

Feature Selection

For reasons of both efficiency and efficacy, feature selection is often used when applying machine learning methods to text categorization. We reduce the feature space by eliminating words that appear in only a single document (hapax legomena), then selecting the 1000 words with highest mutual information with each category. Yang and Pedersen [28] compared a number of methods for feature selection. We used a mutual information measure (Cover and Thomas [5]) that is similar to their information gain measure. The mutual information MI(F, C) between a feature, F, and a category, C, is defined as:

$$MI(F,C) = \sum_{F \in \{f, \bar{f}\}} \sum_{C \in \{c, \bar{c}\}} P(F,C) \log \frac{P(F,C)}{P(F)P(C)}$$

We compute the mutual information between every pair of features and categories.

For the non-hierarchical case, we select the 1000 features with the largest MI for each of the 150 categories (vs. all the other categories). For the hierarchical case, we select 1000 features with the largest MI for each of the 13 top-level categories, and also select the 1000 features for each of the 150 second-level categories (vs. the categories that share the same top-level category). We did not rigorously explore the optimum number of features for this problem, but these numbers provided good results on a training validation set so they were used for testing. In all cases, the selected features are used as input to the SVM algorithm that learns the optimum weights for the features, $\vec{w}$.

SVM Parameters

In addition to varying the number of features, SVM performance is governed by two parameters, C (the penalty imposed on training examples that fall on the wrong side of the decision boundary), and p (the decision threshold). The default C parameter value (0.01) was used.

The decision threshold, p, can be set to control precision and recall for different tasks. Increasing p, results in fewer test items meeting the criterion, and this usually increases precision but decreases recall. Conversely, decreasing p typically decreases precision but increases recall. We chose p so as to optimize performance on the F_1 measure on a training validation set. For the flat non-hierarchical models, p=0.5. For the hierarchical models, p=0.2 for the multiplicative decision rule, and p_1=0.2 (first level) and p_2=0.5 (second level) for the Boolean decision rule.

RESULTS

As just described decision thresholds were established on a training validation set. For each category, if a test item exceeds the decision threshold, it is judged to be in the category. A test item can be in zero, one, or more than one categories. From this we compute precision (P) and recall (R). These are micro-averaged to weight the contribution of each category by the number of test examples in it. We used the F measure to summarize the effects of both precision and recall. With equal weights assigned to precision and recall, $F_1 = 2*P*R/(P+R)$.

For each test example, we compute the probability of it being in each of the 13 top-level categories and each of the 150 second-level categories. Recall that for the non-hierarchical case, models were learned (and features chosen) to distinguish each category from all other categories, and that for the hierarchical case, models were learned (and features chosen) to distinguish each category from only those categories within the same top-level category. We explored two general ways to combine probabilities from the first and second level for the hierarchical approach. In one case we first set a threshold at the top level and only match second-level categories that pass this test – that is we compute a Boolean function $P(L1)\&\&P(L2)$. In order to be correctly classified, a test instance must satisfy both constraints. This method is quite efficient, since large numbers of second level categories do not need to be tested. In the other case, we compute $P(L1)*P(L2)$. This approach allows matches even though the scores at one level fall below threshold. Although the non-hierarchical models are not trained to use top-level information, we can compute the same probabilities for this case as well.

Learned Features

There are many differences in the learned features for the hierarchical and non-hierarchical models. For example, the feature "sports" is useful in distinguishing the top-level category Sports & Recreation from the other top-level categories. The feature "sports" is also useful for distinguishing between some second-level categories -- it has a high weight for News & Magazines (under Society & Politics), SportingGoods (under Shopping & Services), and Collecting (under Hobbies & Interests). But, the feature "sports" is not selected for any categories using the flat non-hierarchical approach. It is difficult to study this systematically, but the different learning approaches are serving an important role in feature selection and weighting.

F_1 Accuracy

Top Level

The overall F_1 value for the 13 top-level categories is .572. Classifying short summaries of very heterogeneous web pages is a difficult task, so we did not expect to have exceptionally high accuracy. Performance on the original training set is only .649, so this is a difficult learning task and generalization to the test set is quite reasonable. This level of accuracy is sufficient to support some web search tasks [3], and could also be used as an aid to human indexers. In addition, we know that we can improve the absolute level of performance by 15%-20% using the full text of pages, and by optimizing the *C* parameter. What is of more interest to us here is the comparative performance of hierarchical and non-hierarchical approaches.

Second Level, Non-Hierarchical (Baseline)

The overall F_1 value for the 150 second-level categories, treated as a flat non-hierarchical problem, is .476. There is a drop in performance in going from 13 to 150 categories. Performance for the 150 categories is better than the .364 value reported by McCallum et al. [17], but it is difficult to compare precisely because they use average precision not F_1, the categories themselves are different, and they use the full text of pages. The .476 non-hierarchal value will serve as the baseline for looking at the use of hierarchical models.

Performance varies widely across categories. The five categories with highest and lowest F_1 scores (based on at least 30 test instances) are shown in Table 2.

The most difficult categories to learn models for are those based, at least in part, on non-content distinctions. It might be useful to learn models for genre (e.g., for kids, magazines, web) in addition to our current use of content-based models.

Second Level, Hierarchical

As described above, we examined both a multiplicative scoring function, *P(L1)*P(L2)*, and a Boolean scoring function, *P(L1)&&P(L2)*. The former allows for matches that fall below individual thresholds, and the latter requires that the threshold be exceeded at every level. Both scoring rules allow items to be classified into multiple classes.

The overall F_1 value for the *P(L1)*P(L2)* scoring function is .495, at the threshold of p=0.20 established on the validation set. This represents a 4% improvement over the baseline flat models, and is statistically significant using a sign test, p<.01. The overall F_1 value for the *P(L1)&&P(L2)* scoring function is .497, at the thresholds of p_1=0.20 and p_2=0.50 established on the validation set. This again is a small improvement over the non-hierarchical model, and is significantly different than the baseline using a sign test, p<0.01. Somewhat surprisingly, there is no difference between the sequential Boolean rule and the multiplicative scoring rule. The added efficiencies that can be obtained with the hierarchical model and the Boolean rule make it a good overall choice, as we discuss in more detail below.

An upper bound on the performance of these hierarchical models can be obtained by assuming that the top-level classification is correct (i.e., *P(L1)=1*). In this case, performance is quite good with an F_1 value of .711, so there is considerable room for improvement.

	Category Name
Best F1	
0.841	*Health & Fitness/Drugs & Medicines*
0.797	*Home & Family/Real Estate*
0.781	*Reference & Education/K-12 Education*
0.750	*Sports & Recreation/Fishing*
0.741	*Reference & Education/Higher & Cont. Ed.*
Worst F1	
0.034	*Society & Politics/World Cultures*
0.088	*Home & Family/For Kids*
0.122	*Computers & Internet/News & Magazines*
0.131	*Computers & Internet/Internet & the Web*
0.133	*Business & Finance/Business Professions*

Table 2 - Best and worst F1 score by category

Efficiency

Offline training

As noted earlier, linear SVM models can be learned quite efficiently with Platt's SMO algorithm. The total training time for all the models described in the paper was only about 30 minutes. Training time for all the non-hierarchical models on 50078 examples in each of 150 categories was 729 CPU seconds. Training time for all 150 hierarchical second-level models was 128 CPU seconds. Training is faster here because the negative set is smaller, including only items in the same top-level category. Training time for the 13 top-level categories was 1258 CPU seconds. All times were computed on a standard 266MHz Pentium II PC running Windows NT.

Online evaluation

At run time we need to take the dot product of the learned weight vector for each category and the feature vector for the test item. The cost of this operation depends on the number of features and the number of categories.

Efficiency can be improved if some category comparisons can be omitted without hurting accuracy. As we have seen above, the Boolean decision function accomplishes this with no decrease in classification accuracy. For the non-hierarchical model, each test instance must be compared with all 150 second-level categories. For the hierarchical model with the multiplicative scoring rule, each test instance must be compared to all 13 first-level categories and all 150 second-level categories. For the Boolean scoring rule, each test instance must be compared to all 13 first-level categories but only to second-level categories that pass the first-level criterion. For the top-level threshold value of p=0.20, only 7.4% of the second-level categories need to be examined. Top-level comparisons are still required for the Boolean decision rule, so overall 14.8% of the comparisons used for the multiplicative rule and 16.1% of the comparisons used for the non-hierarchical model are required, both representing a considerable savings at evaluation time.

CONCLUSION

The research described in this paper explores the use of hierarchical structure for classifying a large, heterogeneous collection of web content to support classification of search results. We used SVMs, which have been found to be an efficient and effective learning method for text classification, but not previously explored for hierarchical problems. We use the hierarchical structure for two purposes. First, we train second-level category models using different contrast sets (either within the same top-level category in the hierarchical case, or across all categories in the non-hierarchical case). Second, we combine scores from the top- and second-level models using different combination rules, some requiring a threshold to be exceeded at the top level before comparisons at the second level are made.

We found small advantages in the F_1 accuracy score for the hierarchical models, compared with a baseline flat non-hierarchical model. We found no difference between a multiplicative scoring function and a sequential Boolean function for combining scores from the two levels of the hierarchy. Since the sequential Boolean approach is much more efficient, requiring only 14%-16% of the number of comparisons, we find it to be a good choice.

There are a number of interesting directions for future research. We should be able to obtain further advantages in efficiency in the hierarchical approach by reducing the number of features needed to discriminate between categories within the same top-level category. Koller and Sahami [13] have shown that one can dramatically reduce the number of features for simple Reuters categories without decreasing accuracy too much, and this is definitely worth trying with our content and learning algorithm. There are some preliminary indications that this will work in our application. The SVM model assigns a weight of 0 to features that are not predictive (and could thus be omitted). We find a larger number of features with 0 weights in the hierarchical case (137 per 1000) than the non-hierarchical case (85 per 1000). There are also many near-zero weights that contribute little to the overall score. Varying the number of features for further efficiency gains seems promising to try with our content and learning model.

The sequential decision model provides large efficiency gains, so it will be important to see how it generalizes to more than two levels, where error cascading may be more of an issue. There are at least two ways to address the problem. One involves improved classifiers and methods for setting appropriate decisions thresholds at multiple levels. Another approach is through the use of interactive interfaces so that people are involved in making some of the critical decisions. This is a rich space for further exploration.

Our research adds to a growing body of work exploring how hierarchical structures can be used to improve the efficiency and efficacy of text classification. We worked with a large, heterogeneous collection of web content -- a scenario that is increasingly characteristic of the information management tasks that individuals and organizations face. And, we successfully extended SVM models to an application that takes advantage of hierarchical structure, for both category learning and run time efficiencies.

ACKNOWLEDGMENTS

We are grateful to John Platt for help with the Support Vector Machine code, and to four anonymous reviewers for their comments.

REFERENCES

1. Apte, C., Damerau, F. and Weiss, S. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems, 12(3),* 233-251, 1994.
2. Chakrabarti, S., Dom, B., Agrawal, R. and Raghavan, P. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal 7,* 163-178, 1998.
3. Chen, H. and Dumais, S. Bringing order to the web: Automatically categorizing search results. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'00),* 145-152, 2000.
4. Cohen, W.W. and Singer, Y. Context-sensitive learning methods for text categorization *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96),* 307-315, 1996.
5. Cover, T. and Thomas, J. *Elements of Information Theory.* Wiley, 1991.
6. D'Alessio, S., Murray, M., Schiaffino, R. and Kershenbaum, A. Category levels in hierarchical text

categorization. *Proceedings of EMNLP-3, 3rd Conference on Empirical Methods in Natural Language Processing*, 1998.

7. Dumais, S. T., Platt, J., Heckerman, D. and Sahami, M. Inductive learning algorithms and representations for text categorization. *Proceedings of the Seventh International Conference on Information and Knowledge Management (CIKM'98)*, 148-155, 1998.
8. Fuhr, N., Hartmanna, S., Lustig, G., Schwantner, M., and Tzeras, K. Air/X – A rule-based multi-stage indexing system for lage subject fields. *Proceedings of RIAO'91*, 606-623, 1991.
9. Hayes, P.J. and Weinstein, S.P. CONSTRUE: A System for Content-Based Indexing of a Database of News Stories. *Second Annual Conference on Innovative Applications of Artificial Intelligence*, 1990.
10. Hearst, M., and Karadi, C. Searching and browsing text collections with large category hierarchies. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'97), Conference Companion*, 1997.
11. Hearst, M. and Pedersen, J. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. *Proceedings of 19th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, 1996.
12. Joachims, T. Text categorization with support vector machines: Learning with many relevant features. *Proceedings of European Conference on Machine Learning (ECML'98)*, 1998
13. Koller, D. and Sahami, M. 1997. Hierarchically classifying documents using very few words. *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, 170-178, 1997.
14. Landauer, T., Egan, D., Remde, J., Lesk, M., Lochbaum, C., and Ketchum, D. Enhancing the usability of text through computer delivery and formative evaluation: The SuperBook project. *Hypertext – A Psychological Perspective*. Ellis Horwood, 1993.
15. Larkey, L. Some issues in the automatic classification of U.S. patents. In *Working Notes for the AAAI-98 Workshop on Learning for Text Categorization*, 1998.
16. Lewis, D.D. and Ringuette, M.. A comparison of two learning algorithms for text categorization. *Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94)*, 81-93, 1994.
17. McCallum, A., Rosenfeld, R., Mitchell, T. and Ng, A. Improving text classification by shrinkage in a hierarchy of classes. *Proceedings of the Fifteenth International Conference on Machine Learning, (ICML-98)*, 359-367, 1998.
18. Mladenic, D. and Grobelnik, M. Feature selection for classification based on text hierarchy. *Proceedings of the Workshop on Learning from Text and the Web*, 1998.
19. Ng, H.T., Goh, W.B. and Low, K.L, *Proceedings of 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, 67-73, 1997.
20. Platt, J. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods –Support Vector Learning*. B. Schölkopf, C. Burges, and A. Smola, eds., MIT Press, 1999.
21. Ruiz, M.E. and Srinivasan, P. Hierarchical neural networks for text categorization. *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, 281-282, 1999.
22. Schütze, H., Hull, D. and Pedersen, J.O. A comparison of classifiers and document representations for the routing problem. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, 229-237, 1995.
23. Vapnik, V., *Estimation of Dependencies Based on Data [in Russian]*, Nauka, Moscow, 1979. (English translation: Springer Verlag, 1982.)
24. Vapnik, V., *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.
25. Weigend, A.S., Wiener, E.D. and Pedersen, J.O. Exploiting hierarchy in text categorization. *Information Retrieval, 1(3)*, 193-216, 1999.
26. Yang, Y. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, 13-22, 1994.
27. Yang, Y. and Lui, Y. A re-examination of text categorization methods. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, 42-49, 1999.
28. Yang, Y. and Pedersen, J.O. A comparative study on feature selection in text categorization. *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, 412-420, 1997.
29. Zamir, O. and Etzioni, O. Web document clustering: A feasibility demonstration. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, 46-54, 1998.
30. http://www.looksmart.com

EXHIBIT B

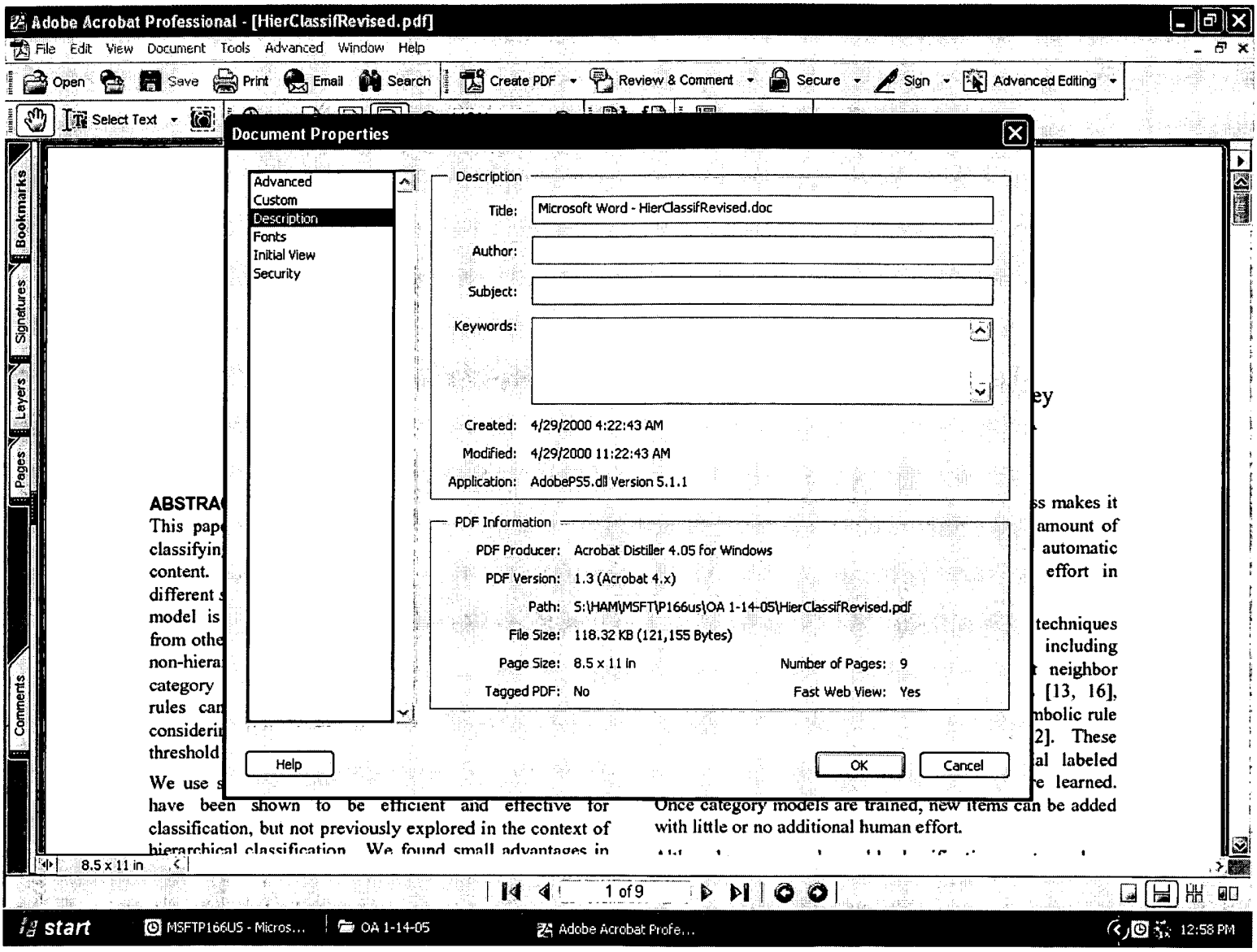

Adobe Acrobat Professional - [HierClassifRevised.pdf]
File Edit View Document Tools Advanced Window Help
Open Save Print Email Search Create PDF Review & Comment Secure Sign Advanced Editing
Select Text
Document Properties
Advanced
Custom
Description
Fonts
Initial View
Security
Description
Title: Microsoft Word - HierClassifRevised.doc
Author:
Subject:
Keywords:
Created: 4/29/2000 4:22:43 AM
Modified: 4/29/2000 11:22:43 AM
Application: AdobePS5.dll Version 5.1.1
PDF Information
PDF Producer: Acrobat Distiller 4.05 for Windows
PDF Version: 1.3 (Acrobat 4.x)
Path: S:\HAM\MSFT\P166us\OA 1-14-05\HierClassifRevised.pdf
File Size: 118.32 KB (121,155 Bytes)
Page Size: 8.5 x 11 in
Number of Pages: 9
Tagged PDF: No
Fast Web View: Yes
Help
OK
Cancel
Bookmarks
Signatures
Layers
Pages
Comments
8.5 x 11 in
1 of 9
start
MSFTP166US - Micros...
OA 1-14-05
Adobe Acrobat Profe...
12:58 PM

EXHIBIT C

Bringing Order to the Web: Automatically Categorizing Search Results

Hao Chen
School of Information Management & Systems
University of California
Berkeley, CA 94720 USA
hchen@sims.berkeley.edu

Susan Dumais
Microsoft Research
One Microsoft Way
Redmond, WA 99802 USA
sdumais@microsoft.com

ABSTRACT

We developed a user interface that organizes Web search results into hierarchical categories. Text classification algorithms were used to automatically classify arbitrary search results into an existing category structure on-the-fly. A user study compared our new category interface with the typical ranked list interface of search results. The study showed that the category interface is superior both in objective and subjective measures. Subjects liked the category interface much better than the list interface, and they were 50% faster at finding information that was organized into categories. Organizing search results allows users to focus on items in categories of interest rather than having to browse through all the results sequentially.

Keywords

User Interface, World Wide Web, Search, User Study, Text Categorization, Classification, Support Vector Machine

INTRODUCTION

With the exponential growth of the Internet, it has become more and more difficult to find information. *Web search* services such as AltaVista, InfoSeek, and MSNWebSearch were introduced to help people find information on the web. Most of these systems return a ranked list of web pages in response to a user's search request. Web pages on different topics or different aspects of the same topic are mixed together in the returned list. The user has to sift through a long list to locate pages of interest. Since the 19th century, librarians have used classification systems like Dewey and Library of Congress classification to organize vast amounts of information. More recently, *Web directories* such as Yahoo! and LookSmart have been used to classify Web pages. The manual nature of the directory compiling process makes it impossible to have as broad coverage as the search engines, or to apply the same structure to intranet or local files without additional manual effort.

To combine the advantage of structured topic information in directories and broad coverage in search engines, we built a system that takes the web pages returned by a search engine and classifies them into a known hierarchical structure such as LookSmart's Web directory [24]. The system consists of two main components: 1) a text classifier that categorizes web pages on-the-fly, and 2) a user interface that presents the web pages within the category structure and allows the user to manipulate the structured view (Figure 1).

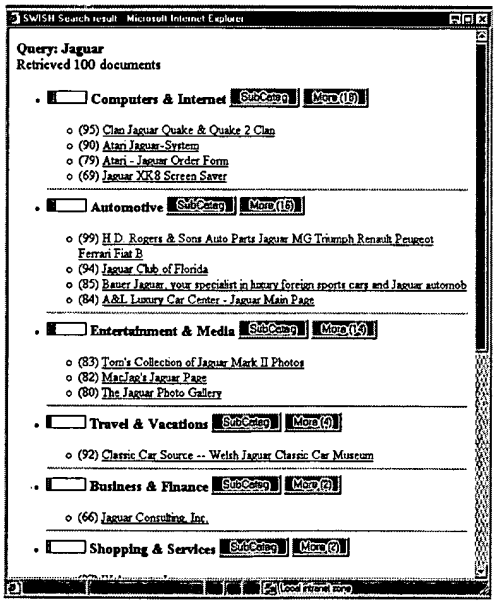


Figure 1: Presenting web pages within category structure

RELATED WORK

Generating structure

Three general techniques have been used to organize documents into topical contexts. The first one uses *structural information* (meta data) associated with each document. The DynaCat system by Pratt [15] used meta data from the UMLS medical thesaurus to organize search results. Two prototypes developed by Allen [1] used meta

data from the Dewey Decimal System for organizing results. In the SuperBook project [10], paragraphs of texts were organized into an author-created hierarchical table of contents. Marchionini et al. [12] also used table of content views for structuring information from searches in the Library of Congress digital library. Others have used the link structure of Web pages to automatically generate structured views of Web sites. Maarek et al.'s WebCutter system [11] displayed a site map tailored to the user's search query. Wittenburg and Sigman's AMIT system [18] showed search results in the context of an automatically derived Web site structure. Chen et al.'s Cha-Cha system [4] also organized search results into automatically derived site structures using the shortest path from the root to the retrieved page. Manually-created systems are quite useful but require a lot of initial effort to create and are difficult to maintain. Automatically-derived structures often result in heterogeneous criteria for category membership and can be difficult to understand.

A second way to organize documents is by *clustering*. Documents are organized into groups based their overall similarity to one another. Zamir et al. [19, 20] grouped Web search results using suffix tree clustering. Hearst et al. [7, 8] used the scatter/gather technique to organize and browse documents. One problem with organizing search results in this way is the time required for on-line clustering algorithms. Single-link and group-average methods typically take $O(n^2)$ time, while complete-link methods typically take $O(n^3)$, where n is the number of documents returned. Linear-time algorithms like k-means are more efficient being $O(nkT)$, where k is the number of clusters and T the number of iterations. In addition, it is difficult to describe the resulting clusters to users. Clusters are usually labeled by common phrases extracted from member documents, but it is often difficult to quickly understand the contents of a cluster from its label.

A third way to organize documents is by *classification*. In this approach, statistical techniques are used to learn a model based on a labeled set of training documents (documents with category labels). The model is then applied to new documents (documents without category labels) to determine their categories. Chakrabarti et al. [2], Chekuri [3], and Mladenic [13] have developed automatic classifiers for subsets of pages from the Yahoo! Web directory. Only a small number of high level categories were used in their published results. And, the focus of these papers was on the underlying text classification algorithms and not on user interfaces that exploit the results. Recently, Inktomi [22] announced that it had developed techniques for automatic classification of web pages. However, its technical details were not disclosed and we are not aware of any search services employing this technology.

Using structure to support search

A number of web search services use category information to organize the search results. Yahoo! [27], Snap [26] and LookSmart [24] show the category label associated with each retrieved page. Results are still shown as a ranked list with grouping occurring only at the lowest level of the hierarchy (for Yahoo! and Snap). There is, for example, no way to know that 70% of the matches fell into a single top-level category. In addition, these systems require pre-tagged content. Before any new content can be used, it must be categorized by hand. Northern Light [25] provides Custom Folders in which the retrieved documents are organized hierarchically. The folders are organized according to several dimensions -- source (sites, domains), type (personal page, product review), language, and subject. Individual categories can be explored one at a time. But, again no global information is provided about the distribution of search results across categories.

The most common interface for manipulating hierarchical category structures is a hierarchical tree control, but other techniques have been explored as well. Johnson et al. [9] used a treemap that partitioned the display into rectangular bounding boxes representing the tree structure. Characteristics of the categories and their relationships were indicated by their sizes, shapes, colors, and relative positions. Shneiderman et al. [17] have recently developed a two-dimensional category display that uses categorical and hierarchical axes, called hieraxes, for showing large results sets in the context of categories. Hearst et al. [5] used three-dimensional graphics to display categories together with their documents. Multiple categories could be displayed simultaneously along with their hierarchical context. In all of these systems, documents must have pre-assigned category tags.

Few studies have evaluated the effectiveness of different interfaces for structuring information. Landauer et al. [10] compared two search interfaces for accessing chemistry information -- SuperBook which used a hierarchical table of contents, and PixLook which used a traditional ranked list. Browsing accuracy was higher for SuperBook than PixLook. Search accuracy and search times were the same for the two interfaces. However, different text pre-processing and search algorithms were used in the two systems so it is difficult to compare precisely. More recently, Pratt et al. [16] compared DynaCat, a tool that automatically categorized results using knowledge of query types and a model of domain terminology, with a ranked list and clustering. Subjects liked DynaCat's category organization of search results. Subjects found somewhat more new answers using DynaCat, but the results were not reliable statistically, presumably because there were only 15 subjects and 3 queries in the experiment.

In this paper we describe a new system showing how automatic text classification techniques can be used to

organize search results. A statistical text classification model is trained offline on a representative sample of Web pages with known category labels. At query time, new search results are quickly classified on-the-fly into the learned category structure. This approach has the benefit of using known and consistent category labels, while easily incorporating new items into the structure. The user interface compactly displays web pages in a hierarchical category structure. Heuristics are used to order categories and select results within categories for display. Users can further expand categories on demand. Tooltip-like overlays are used to convey additional information about individual web pages or categories on demand. We compared our category interface with a traditional list interface under exactly the same search conditions. We now describe each of these components in more detail.

TEXT CLASSIFICATION

Text classification involves a training phase and a testing phase. During the training phase, web pages with known category labels are used to train a classifier. During the testing or operational phase, the learned classifier is used to categorize or tag new web pages.

Data Set

For training purposes, we used a collection of web pages from LookSmart's Web directory [24]. LookSmart's directory is created and maintainted by 180 professional Web editors. For our experiments, we used the directory as it existed in May 1999. At that time there were 13 top-level categories, 150 second-level categories, and over 17,000 categories in total. On average each web page was classified into 1.2 categories.

Pre-processing

A text pre-processing module extracted plain text from each web page. In addition, the title, description, keyword, and image tag fields were also extracted if they existed. A vector was created for each page indicating which terms appeared in that page.

The results returned by search engines contain a short summary of information about each result. Although it is possible to download the entire contents of each web page, it is too time consuming to be applicable in a networked environment. Therefore, in our prototype, the initial training and subsequent classification are performed using only summaries of each web page. The training summaries were created using the title, the keyword tag, and either the description tag if it existed or the first 40 words otherwise. When classifying search results we use the summary provided in the search results.

Classification

A Support Vector Machine (SVM) algorithm was used as the classifier, because it has been shown in previous work to be both very fast and effective for text classification problems [5][14]. Roughly speaking, a linear SVM is a hyperplane that separates a set of positive examples (i.e., pages in a category) from a set of negative examples (i.e., pages not in the category). The SVM algorithm maximizes the margin between the two classes; other popular learning algorithms minimize different objective functions like the sum of squared errors. Web pages were pre-processed as described above. For each category we used the 1000 terms that were most predictive of the category as features. Vectors for positive and negative examples were input into the SVM learning algorithm. The resulting SVM model for each category is a vector of 1000 terms and associated weights that define the hyperplane for that category.

We used 13,352 pre-classified web pages to train the model for the 13 top-level categories, and between 1,985 and 10,431 examples for each of these categories to train the appropriate second-level category models. The total time to learn all 13 top-level categories and 150 second-level categories was only a few hours. Once the categories are learned, the results from any user query can be classified. At query time, each page summary returned by the search engine is compared to the 13 top-level category models. A page is placed into one or more categories, if it exceeds a pre-determined threshold for category membership. Pages are classified into second-level categories only on demand using the same procedure.

We explored a number of parameter settings and text representations and used the optimal ones for classification in our experiment. Our fully automatic methods for assigning category labels agreed with the human-assigned labels almost 70% of the time. Most of the disagreements were because additional labels were assigned (in addition to the correct one), or no labels were assigned. This is good accuracy given that we were working with only short summaries and very heterogeneous web content. Although classification accuracy is not perfect, we believe it can still be useful for organizing Web search results.

USER INTERFACE

The search interface accepted query keywords, passed them to a search engine selected by the user, and parsed the returned pages. Each page was classified into one or more categories using the learned SVM classifier. The search results were organized into hierarchical categories as shown in Figure 1. Under each category, web pages belonging to that category were listed. The category could be expanded (or collapsed) on demand by the user. To save screen space, only the title of each page was shown (the summary can be viewed by hover text, to be discussed later). Clicking on the title hyperlink brought up the full content of the web page in another browser window, so that the category structure and the full-text of pages were simultaneously visible.

Information Overlays

There is a constant conflict between the large amount of information we want to present and the limited screen real estate. We presented the most important information (titles of web pages and category labels) as text in the interface, and showed other information using small icons or transient visual overlays. The techniques we used included:

- A partially filled green bar in front of each category label showed the percentage of documents falling into the category. This provided users with an overview of the distribution of matches across categories.
- We presented additional category information (parent and child category labels) as hover text when the mouse hovered over a category title. This allowed users to see the subcategories for category as well as the higher-level context for each page.
- The summaries of the web pages returned by search engines provide users with additional information about the page helping them decide which pages to explore in greater depth. In order to present category context along with the search results, we displayed only titles by default and showed summaries as hover text when the mouse hovered over the titles of web pages.

Distilled Information Display

Even with the help of information overlays, there is still more information than a single screen can accommodate. We developed heuristics to selectively present a small portion of the most useful information on the first screen. The first screen is so important that it usually determines whether the user will continue working on this search or abandon it all together. We wanted to enable the user to either find the information there or identify a path for further exploration. In order to do this effectively we must decide: how many categories to present, how many pages to present in each category, how to rank pages within a category, and how to rank categories.

We presented only top-level categories on the first screen. There were several reasons for this. First, the small number of top level categories helped the user identify domains of interest quickly. Second, it saved a lot of screen space. Third, classification accuracy was usually higher in top level categories. Fourth, it was computationally faster to match only the top-level categories. Fifth, subcategories did not help much when there were only a few pages in the category. The user can expand any category into subcategories by clicking a button.

In each category, we showed only a subset of pages in that category. We decided to show a fixed number of pages (20) across all categories, and divided them in proportion to the number of pages in that category. So, if one category contained 50% of results, we would show 10 pages from that category in the initial view. The user can see all pages in a category by clicking a button.

Three parameters affected how pages are ordered within a category: its original ranking order in the results, its match score (if returned by the search engine), and the probability that it belongs to the category according to the classifier. For the experiment, we used only the rank order in the original search results to determine the order of items within each category. Thus if all the search result fall into one category the category organization returns the same items in the same order as the ranked list.

The categories can be ordered either in a static alphabetical order, or dynamically according to some importance score. The advantage of dynamic ranking is to present the most likely category first. The disadvantage is that it prevents the user from establishing a mental model of the relative position of each category in the browser window. For our experiment, importance was determined by the number of pages in the category. The category with the most items in it was shown first, and so on.

USER STUDY

A user study was conducted to compare the category-based interface (referred to as "Category Interface" henceforth) with the conventional search interface where pages are arranged in a ranked list (referred to as "List Interface" henceforth). The two interfaces are shown in Figure 2.

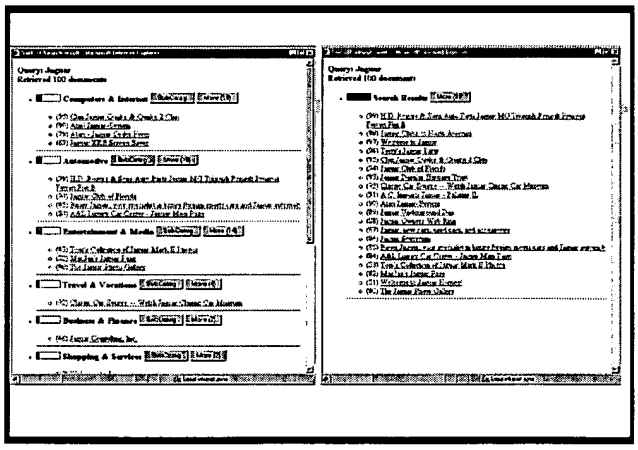


Figure 2: Category vs. List Interface

The top 100 search results for the query *"jaguar"* are used in this example. Twenty items are shown initially in both interfaces. In the List interface the 20 items can be seen without scrolling; in the Category interface scrolling is always required in spite of our attempt to conserve screen space. In both interfaces, summaries are shown on hover. Both interfaces contain a ShowMore button which is used to show the remaining items in the category; in the case of the List interface the remaining 80 items are shown. In addition, in the Category interface a SubCategory button is

used to sub-categorize the pages within that category. The same control program is used in both cases, so timing is the same in both interfaces.

Methods

Subjects

Eighteen subjects of intermediate web ability participated in the experiment. Subjects were adult residents of the Seattle area recruited by the Microsoft usability lab, and represent a range of ages, backgrounds, jobs and education level.

Procedure

The experiment was divided into two sessions with a voluntary break between. Subjects used the Category interface in one session and the List interface in the other. The user read a short tutorial before each session began. During each session, the user performed 15 web search tasks, for a total of 30 search tasks. At the end of the experiment, the user completed an online questionnaire giving his/her subjective rating of the two interfaces. The total time for the experiment was about 2 hours.

During the experiment, the subject worked with three windows (Figure 3). The control window on the top shows the task and the query keywords. In this example, the ***task*** is to find out about *"renting a Jaguar car"* and the ***query*** we automatically issued is *"jaguar"*. The search results were displayed in the left bottom window. In the Category interface, the results were automatically organized into different categories, and in the List interface, the top 20 items were shown on the initial screen.

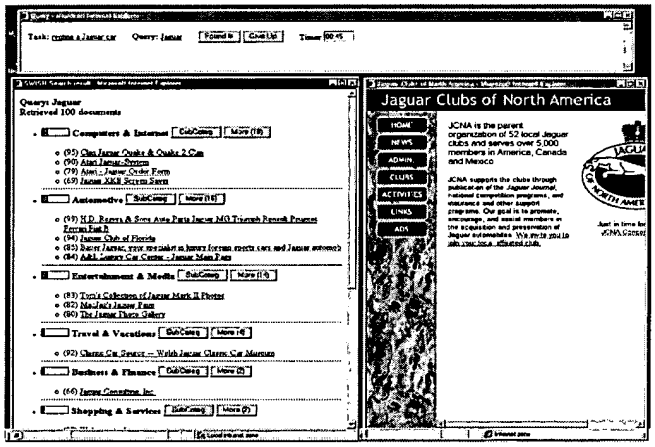


Figure 3: Screen of the User Study

When the subject clicked on a hyperlink, the page opened in the right window. When the subject found an answer, s/he clicked on the "Found It!" button in the control window. If no answer could be found, s/he clicked on the "Give Up" button. There was a timer in the control window that reminded the subject after five minutes had passed. If a reminder occurred, the subject could continue searching or move on to the next task. User events such as hovering over a hyperlink to read the summary, clicking on a hyperlink to read the page, expanding or collapsing the list were logged.

Search Tasks

The 30 search tasks were selected from a broad range of topics, including sports, movies, travel, news, computers, literature, automotive, local interest, etc. Ten of the queries were popular queries from users of MSNWebSearch. In order to facilitate evaluation we selected tasks that had reasonably unambiguous answers in the top 100 returned pages (a kind of known-item search). The tasks varied in difficulty – 17 had answers in the top 20 items returned (on the first page in the List interface), and 13 had answers between ranks 21 and 100. The tasks also varied in how much manipulation was required in the Category interface – 10 required subjects to use ShowMore or SubCategory expansion, and 10 required some scrolling because the correct category was not near the top.

To ensure that results from different subjects were comparable, we fixed the keywords for each query in the experiment. We also cached the search results before the experiments so that each subject got the same results for the same query. The MSNWebSearch engine [22] was used to generate the search results.

Each subject performed the same 30 search tasks. For 15 tasks they used the Category interface and for 15 they used the List interface. The order in which queries were presented and whether the Category or List interface was used first was counterbalanced across subjects. Nine lists of tasks were used -- each list contained all the tasks in a different order and was assigned to a pair of subjects, one in the Category-first condition and one in the List-first condition. This yoking of presentation orders reduces error variance which is desirable given the relatively small number of subjects and tasks we used.

Results

The main independent variable is the Category interface vs. the List interface. The order of presentation (List first or Category first) is a between subject variable. We analyzed both subjective questionnaire measures and objective measures (search time, accuracy, and interactions with the interface such as hovering, and displaying Web pages).

Subjective questionnaire measures

After the experiment, subjects completed a brief online questionnaire. The questionnaire covered prior experience with Web searching, ratings of the two interfaces (on a 7-point scale), and open-ended questions about the best and worst aspects of each interface. Seventeen of the eighteen subjects used the Web at least every week, and eleven of the eighteen subjects searched for information on the Web at least every week. The most popular Web search service among our subjects was Yahoo!.

Subjects reported that the Category interface was "easy to use" (6.4 vs. 3.9, $t(17) = 6.41$; $p<<0.001$), they "liked using it" (6.7 vs. 4.3, $t(17) = 6.01$; $p<<0.001$), they were "confident that I could find the information if it was there" (6.3 vs. 4.4, $t(17) = 4.91$; $p<<0.001$), that it was "easy to get a good sense of the range of alternatives" (6.4 vs. 4.2, $t(17) = 6.22$; $p<<0.001$), and that they "prefer this to my usual search engine" (6.4 vs. 4.3, $t(17) = 4.13$; $p<<0.001$). On all of our overall measures subjects much preferred the Category interface.

For the two questions that asked about the usefulness of interface features (hover text and ShowMore), there were no reliable differences between interfaces, suggesting that subjects did not simply have an overall positive bias in responding to questions about the Category interface. Subjects thought the display of page summaries in hover text was useful in both interfaces (6.5 Category vs. 6.4 List, $t(17) = 0.36$; $p<0.72$), and that the ShowMore option was useful (6.5 Category vs. 6.1 List, $t(17) = 1.94$; $p<0.07$).

Accuracy/GiveUp.

When creating search tasks, we had a target correct answer in mind. However, other pages might be relevant as well, so we examined all pages that subjects said were relevant to see if they in fact answered the search task. We looked at performance with strict and liberal scoring of accuracy. For strict scoring only pages that were deemed by the experimenters to be relevant (after including additional pages found by subjects that we had missed) were counted as relevant. Using the strict criterion, there were slightly more wrong answers in the List interface (1.72 out of 30) than in the Category interface (1.06 out of 30), but this difference is not reliable statistically using a paired t-test ($t(17) = -1.59$; $p<.13$). The lack of difference between interfaces is not surprising, since it reflects a difference in criterion about what the correct answer is rather than task difficulty per se. For liberal scoring, any answer that subjects said was relevant was deemed relevant, so by definition, there were no wrong answers in either interface. We used the liberal scoring in subsequent analyses.

Subjects were allowed to give up if they could not find an answer. They could do this at any time during a trial. After 5 minutes had elapsed for a task, subjects were notified and encouraged to move onto the next task. Some subjects continued searching, but most gave up at this time. There are significantly more tasks on which subjects gave up in the List interface than in the Category interface ($t(17) = -2.41$; $p<.027$), although the absolute number of failures is small in both interfaces (0.77 in List and 0.33 in Category).

Search Time

We used the median search time across queries for statistical tests, because reaction time distributions are often skewed and statistical tests can be influenced by outliers. (We also find exactly the same results using mean reaction times, so outliers were not a problem in this experiment.) A 2x2 mixed design was used to measure differences in search time. The between subjects factor is whether subjects saw the List or Category interface first, and the within subjects factor is List or Category interface. Median search times are shown in Figure 4.

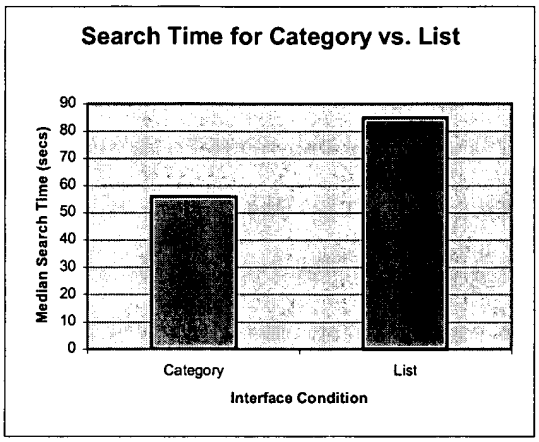


Figure 4: Search time by interface type

There is a reliable main effect of interface type, with a median response time of 56 seconds for the Category interface and 85 seconds for the List interface ($F(1,16) = 12.94$; $p=.002$). The advantage is not due to a speed-accuracy tradeoff or to a tendency to give up on difficult queries, since if anything subjects in the Category interface were more accurate (when scored strictly) and gave up less often. This is a large effect both statistically and practically. It takes subjects 50% longer to find answers using the List interface. On average it took subjects 14 minutes to complete 15 tasks with the Category interface, and 21 minutes with the List interface. There is no effect of the order in which interfaces were shown, list first or category first ($F(1,16) = 0.26$; $p=0.62$). And, there is no interaction between order and interface ($F(1,16) = 1.23$; $p=0.28$), which shows that results are not biased by order of presentation.

There are large individual differences in search time. The fastest subject finished the 30 search tasks in a median of 37 seconds, and the slowest in 142 seconds. But, the advantage of the Category interface is consistent across subjects.

There are also large differences across tasks or queries. The easiest task was completed in a median of 22.5 seconds, and the most difficult task required 166 seconds to complete. We divided the queries into those whose

answers were on the first screen of the List interface (i.e., in the Top20 returned by the search engine) and those whose answers were not in the Top20. The search times are shown in Figure 5. Not surprisingly, there is a reliable main effect of whether the answer is in the Top20 or not -- median time for Top20 (57 seconds) and NotTop20 (98 seconds), $F(1,56) = 16.5$; $p<<.001$.

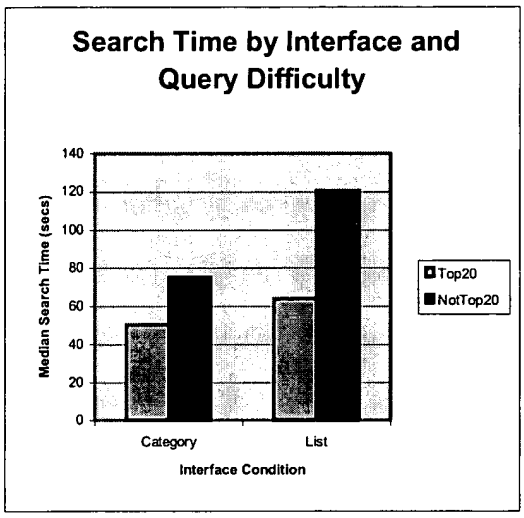


Figure 5: Search time by interface type and query difficulty

There is no interaction between query difficulty and interface ($F(1,56)=2.52$; $p=.12$). The Category interface is beneficial for both easy and hard queries. Although there is a hint that the category interface is more helpful for difficult queries, the interaction is not reliable. The Category interface is still beneficial even when the matching web page is in the first page of results. In our List interface items which were in the Top20 did not require any scrolling, whereas several of the Category interfaces for these items did. The advantage appears to be due to the way in which the category interface breaks the list of returned items down into easily scanable semantic chunks.

Interaction Style – Hovering, Page Views, ShowMore, SubCategory

We measured the number of hovering and page viewing actions subjects performed in the course of finding the answers. Subjects in the List interface hovered on more items than those in the Category interface (4.60 vs. 2.99; $t(17) = -5.54$; $p<<0.001$). The number of pages that subjects actually viewed in the right window is somewhat larger in the List interface (1.41 List vs. 1.23 Category; $t(17) = -2.08$; $p<0.053$). Although the difference is not large, it suggests that the category structure can help disambiguate the summary in the hover text. It is interesting to note that the average number of page views is close to 1, suggesting that users could narrow down their search by reading just the titles and summaries. Subjects read the full pages mostly to confirm what they found in the summary. This significantly reduces search time because the short summaries can be read faster than a full page of text, and there is no network latency for accessing summaries (summaries were stored locally, but retrieving the full-text of the pages required net access).

We also measured the expansion operations that subjects used in searching for information. In the List interface, subjects could expand this list of results by ShowMore. In the Category interface, subjects could ShowMore within each category, or they could break down categories into SubCategories. Overall, subjects in the Category interface used more expansion operations (0.78 ShowMore + SubCategories in Category vs. 0.48 Show More in List; $t(17) = 3.54$; $p<0.003$). So, subjects performed more expansion operations in the Category interface, but the selective nature of the operations (i.e., they applied to only a single category) meant that they were nontheless more efficient overall in finding things.

CONCLUSION

We developed and evaluated a user interface that organizes search results into a hierarchical category structure. Support Vector Machine classifiers were built offline using manually classified web pages. This model was then used to classify new web pages returned from search engines on-the-fly. This approach has the advantage of leveraging known and consistent category information to assist the user in quickly focusing in on task-relevant information. The interface allows users to browse and manipulate categories, and to view documents in the context of the category structure. Only a small portion of the most important and representative information is displayed in the initial screen, and hover text and overlay techniques are used to convey more detailed information on demand. A user study compared the category interface with traditional list interface using the same set of tasks, search engine, and search results. The results convincingly demonstrate that the category interface is superior to the list interface in both subjective and objective measures.

There are many directions for further research. One issue to explore is how the results generalize to other domains and task scenarios. The categories used in our experiment were designed to cover the full range of Web content. Nonetheless, not all user queries will match the category structure to the same extent. Results for some queries may fall entirely within one category (e.g., results for the query *"used parts for Jaguar XJ6L"*, would likely fall entirely within the Automobile category). In such cases, the Category interface (given our current display heuristics) is exactly the same as the List interface, so we are no worse off. Results for other queries may not match any of the categories very well. In our current interface we have a "NotCategorized" group at the bottom. In our experiment

5-40% of the results for each query were NotCategorized, but few of the answers were in the NotCategorized group. We hope to deploy our system more widely to look at this issue by getting a large sample of typical user queries. This would also allow us to explore a wider range of user tasks in addition to the known-item scenario we used.

There are also many interesting issues concerning how best to present concise views of search results in their category contexts. We chose to order categories by the number of matches and within each category to order the pages by search rank. Our text classification algorithms can easily handle thousands of categories, and we may have to move beyond our simple display heuristics for such cases.

ACKNOWLEDGMENTS

We are grateful to John Platt for help with the Support Vector Machine code, to Kirsten Risden for help in setting up the user study, and to reviewers for helpful suggestions.

REFERENCES

1. Allen, R. B., Two digital library interfaces that exploit hierarchical structure. In *Proceedings of DAGS95: Electronic Publishing and the Information Superhighway* (1995).
2. Chakrabarti, S., Dom, B., Agrawal, R., and Raghavan, P. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal 7*, (1998), 163-178.
3. Chekuri, C., Goldwasser, M., Raghavan, P. and Upfal, E. Web search using automated classification. In *Sixth International World Wide Web Conference*, Santa Clara, California, Apr. 1997, Poster POS725.
4. Chen, M., Hearst, M., Hong, J., and Lin, J. Cha-Cha: a system for organizing intranet search results. In *Proceedings of the 2nd USENIX Symposium on Internet Technologies and SYSTEMS (USITS)* (Boulder CO, October 1999) (to appear).
5. Dumais, S. T., Platt, J., Heckerman, D. and Sahami, M. Inductive learning algorithms and representations for text categorization. In *Proceedings of ACM-CIKM98*, Nov. 1998.
6. Hearst, M., and Karadi, C. Searching and browsing text collections with large category hierarchies. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI), Conference Companion* (Atlanta GA, March 1997).
7. Hearst, M., and Pedersen, P. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *Proceedings of 19th Annual International ACM/SIGIR Conference* (Zurich 1996).
8. Hearst, M., Pedersen, J., and Karger, D. Scatter/gather as a tool for the analysis of retrieval results. *Working Notes of the AAAI Fall Symposium on AI Applications in Knowledge Navigation* (Cambridge MA, November 1995).
9. Johnson, B., and Shneiderman, B. Treemaps: a space-filling approach to the visualization of hierarchical information structures. In *Sparks of Innovation in Human-Computer Interaction.* Ablex Publishing Corporation, Norwood NJ, 1993
10. Landauer, T., Egan, D., Remde, J., Lesk, M., Lochbaum, C., and Ketchum, D. Enhancing the usability of text through computer delivery and formative evaluation: the SuperBook project. In *Hypertext – A Psychological Perspective.* Ellis Horwood, 1993.
11. Maarek, Y., Jacovi, M., Shtalhaim, M., Ur, S., Zernik, D., and Ben Shaul, I.Z. WebCutter: a system for dynamic and tailorable site mapping. In *Proceedings of the 6th International World Wide Web Conference* (Santa-Clara CA, April 1997).
12. Marchionini, G., Plaisant, C., and Komlodi, A. Interfaces and tools for the Library of Congress national digital library program. *Information Processing and Management, 34*, 535-555, 1998.
13. Mladenic, D. Turning Yahoo into an automatic web page classifier. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98)* 473-474.
14. Platt, J. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods –Support Vector Learning.* B. Schölkopf, C. Burges, and A. Smola, eds., MIT Press, (1999).
15. Pratt, W. Dynamic organization of search results using the umls. In *American Medical Informatics Association Fall Symposium*, 1997.
16. Pratt, W., Hearst, M. and Fagan, L. A knowledge-based approach to organizing retrieved documents. In *Proceedings of AAAI-99.*
17. Shneiderman, B., Feldman, D. and Rose, A. Visualizing digital library search results with categorical and hierarchical axes. CS-TR-3993, UMIACS-TR-99-12. ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/99-03html/99-03.html
18. Wittenburg, K. and Sigman, E. Integration of browsing, searching and filtering in an applet for information access. In *Proceedings of ACM CHI97: Human Factors in Computing Systems,* (Atlanta GA, March 1997).

19. Zamir, O., and Etzioni, O. Grouper: A dynamic clustering interface to web search results. In *Proceedings of WWW8* (Toronto, Canada, May 1999).

20. Zamir, O., and Etzioni, O. Web document clustering: a feasibility demonstration. In *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, 46-54.

21. http://cha-cha.berkeley.edu/

22. http://search.msn.com/

23. http://www.inktomi.com/new/press/directory.html/

24. http://www.looksmart.com/

25. http://www.northernlight.com/

26. http://www.snap.com/

27. http://www.yahoo.com/

EXHIBIT D

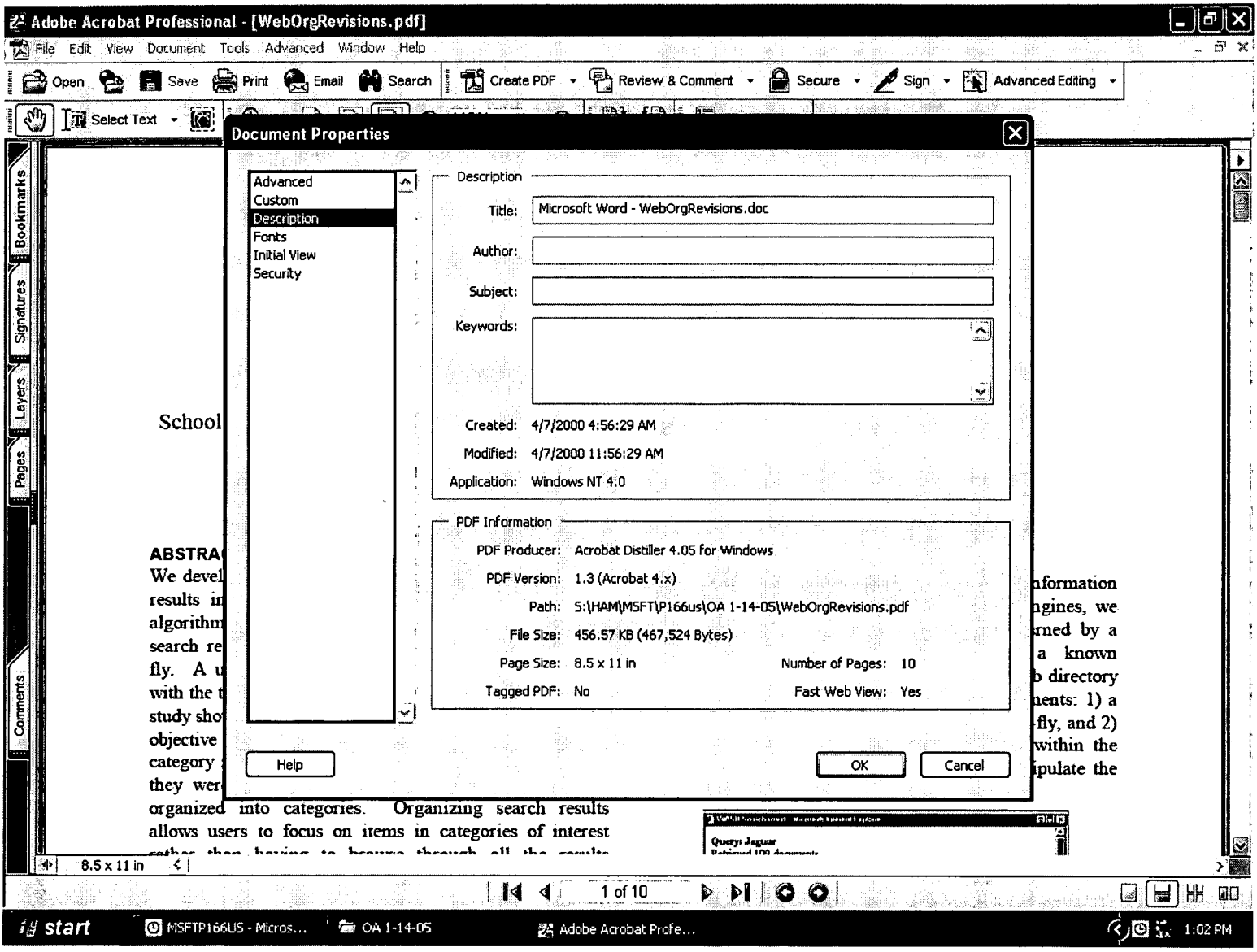

Adobe Acrobat Professional - [WebOrgRevisions.pdf]
File Edit View Document Tools Advanced Window Help
Open Save Print Email Search Create PDF Review & Comment Secure Sign Advanced Editing
Select Text
Document Properties
Advanced
Custom
Description
Fonts
Initial View
Security
Description
Title: Microsoft Word - WebOrgRevisions.doc
Author:
Subject:
Keywords:
Created: 4/7/2000 4:56:29 AM
Modified: 4/7/2000 11:56:29 AM
Application: Windows NT 4.0
PDF Information
PDF Producer: Acrobat Distiller 4.05 for Windows
PDF Version: 1.3 (Acrobat 4.x)
Path: S:\HAM\MSFT\P166us\OA 1-14-05\WebOrgRevisions.pdf
File Size: 456.57 KB (467,524 Bytes)
Page Size: 8.5 x 11 in
Number of Pages: 10
Tagged PDF: No
Fast Web View: Yes
Help
OK
Cancel
Bookmarks
Signatures
Layers
Pages
Comments
School
ABSTRA
We devel
results in
algorithm
search re
fly. A u
with the t
study sho
objective
category
they wer
organized into categories. Organizing search results
allows users to focus on items in categories of interest
information
gines, we
rned by a
a known
directory
nents: 1) a
fly, and 2)
within the
pulate the
8.5 x 11 in
1 of 10
start
MSFTP166US - Micros...
OA 1-14-05
Adobe Acrobat Profe...
1:02 PM

EXHIBIT E

Text Classification for PFS/PSS

Susan Dumais
David Heckerman

Microsoft Research

March 15, 1999

Text Categorization: Road Map

- Overview
  - Text Categorization Basics
  - Inductive Learning Methods
- Results
  - Reuters Results
  - PFS (Preliminary) Results

Text Categorization

- *Text Categorization*: assign objects to one or more of a predefined set of categories using text features
- Example Applications:
  - Sorting new items into existing structures (e.g., general ontologies, file folders, spam vs.not)
  - Information routing/filtering/push
  - Topic specific processing
  - Structured browsing & search

Text Categorization - Methods

- Human classifiers (e.g., Dewey, LCSH, MeSH, Yahoo!, CyberPatrol)
- Hand-crafted knowledge engineered systems (e.g., CONSTRUE)

- Inductive learning of classifiers
  - (Semi-) automatic classification

Classifiers

- A classifier is a function: f(**x**) = conf(class)
  - *from* attribute vectors, **x**=(x_1,x_2, ... x_d)
  - *to* target values, confidence(class)
- Example classifiers
  - if (interest AND rate) OR (quarterly), then confidence(*interest*) = 0.9
  - confidence(*interest*) = 0.3*interest + 0.4*rate + 0.1*quarterly

Inductive Learning Methods

- Supervised learning to build classifiers
  - Labeled training data (i.e., examples of each category)
  - "Learn" classifier
  - Test effectiveness on new instances
- Statistical guarantees of effectiveness
- Classifiers easy to construct and update; requires only subject knowledge
- Customizable for individual's categories and tasks

Text Representation

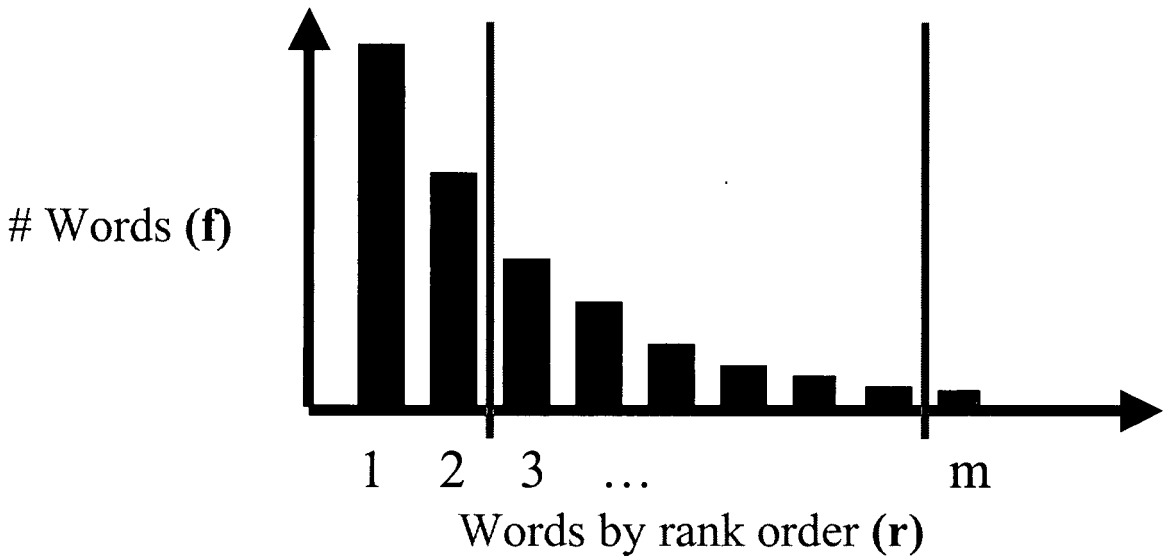
- Vector space representation of documents

	word1	word2	word3	word4 ...
Doc 1 =	<1,	0,	3,	0, ... >
Doc 2 =	<0,	1,	0,	0, ... >
Doc 3 =	<0,	0,	0,	5, ... >

- Text can have 10^7 or more dimensions
  - e.g., 100k web pages had 2.5 million distinct words
- Mostly use: simple words, binary weights, subset of features

Feature Selection

- *Word distribution* - remove frequent and infrequent words based on Zipf's law: log(**f**requency) * log(**r**ank) ~ constant



Feature Selection (cont'd)

- *Fit to categories* - use mutual information to select features which best discriminate category vs. not $MI(x,C) = \sum p(x,C) \log(\frac{p(x,C)}{p(x)p(C)})$
- *Designer features* - domain specific, including non-text features

- Use 100-500 best features from this process as input to learning methods

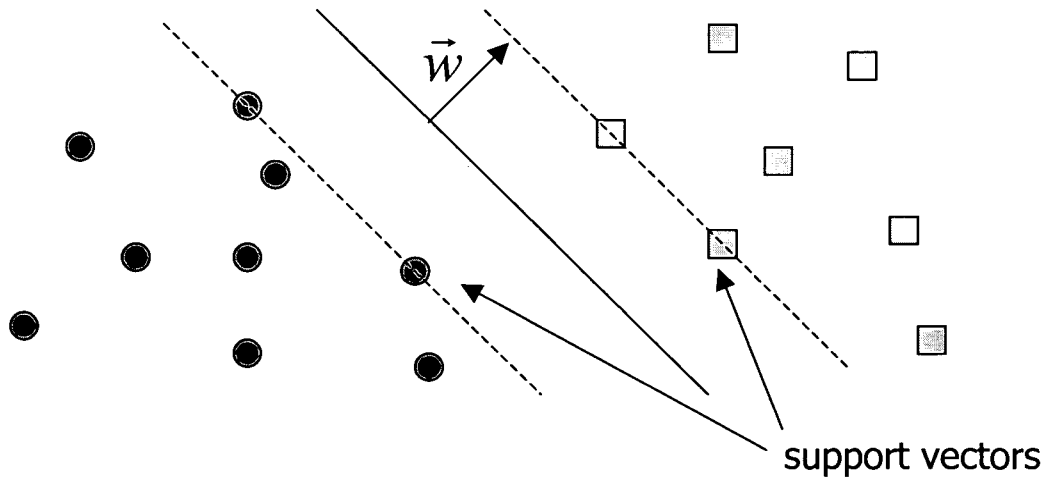
Inductive Learning Methods

- Find Similar
- Decision Trees
- Naïve Bayes
- Bayes Nets
- Support Vector Machines (SVMs)

- All support:
  - Graded category membership - p(category membership); vary threshold depending on task
  - Adaptive - over time; across individuals

Support Vector Machines

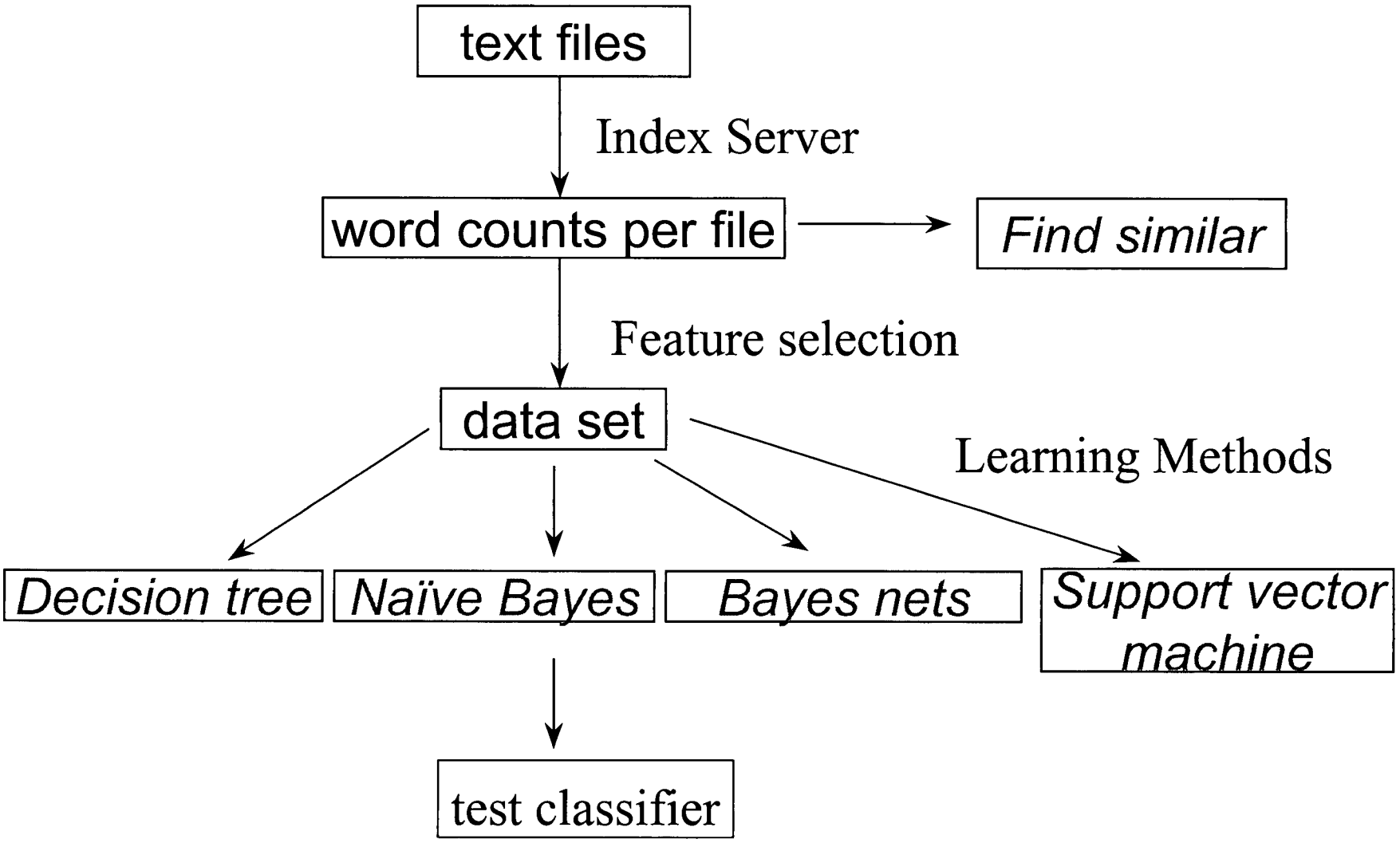
- Vapnik (1979)
- Binary classifiers that maximize margin
  - Find hyperplane separating positive and negative examples
  - Optimization for maximum margin: $\min\|\vec{w}\|^2, \vec{w}\cdot\vec{x}-b\geq 1, \vec{w}\cdot\vec{x}-b\leq -1$
  - Classify new items using: $\vec{w}\cdot\vec{x}$



Support Vector Machines

- Extendable to:
  - Non-separable problems (Cortes & Vapnik, 1995)
  - Non-linear classifiers (Boser et al., 1992)
- Good generalization performance
  - Handwriting recognition (LeCun et al.)
  - Face detection (Osuna et al.)
  - Text classification (Joachims)
- Platt's Sequential Minimal Optimization algorithm very efficient

Text Classification Process



Reuters Data Set (21578 - ModApte split)

- 9603 training articles; 3299 test articles
- Example "interest" article

  2-APR-1987 06:35:19.50
  west-germany
  b f BC-BUNDESBANK-LEAVES-CRE 04-02 0052
  FRANKFURT, March 2
  The Bundesbank left credit policies unchanged after today's regular meeting of its council, a spokesman said in answer to enquiries. The West German discount rate remains at 3.0 pct, and the Lombard emergency financing rate at 5.0 pct.
  REUTER

- Average article 200 words long

Reuters Data Set (21578 - ModApte split)

- 118 categories
  - An article can be in more than one category
  - Learn 118 binary category distinctions
- Most common categories (#train, #test)

- Earn (2877, 1087)
- Acquisitions (1650, 179)
- Money-fx (538, 179)
- Grain (433, 149)
- Crude (389, 189)
- Trade (369,119)
- Interest (347, 131)
- Ship (197, 89)
- Wheat (212, 71)
- Corn (182, 56)

Category: Interest

⌘Example SVM features - $\vec{w}$

- 0.70 prime
- 0.67 rate
- 0.63 interest
- 0.60 rates
- 0.46 discount
- 0.43 bundesbank
- 0.43 baker
- -0.71 dlrs
- -0.35 world
- -0.33 sees
- -0.25 year
- -0.24 group
- -0.24 dlr
- -0.24 january

PFS/PSS - Timing Results

- Training Time - learning models
  - 118 categories; 9603 examples
  - ~ 2 minutes wall time
  - <2 cpu secs/category for the 10 largest categories
- Classification Time
  - 118 categories; 3299 examples
  - <2 msec/document
  - compute $\vec{w} \cdot \vec{x}$
    - x=binary indication of which features present
    - w=learned weight vector
    - sum of << 300 numbers

Accuracy Scores

- Based on contingency table

	Truth: Yes	Truth: No
System: Yes	*a*	*b*
System: No	*c*	*d*

- Effectiveness measure for binary classification
  - error rate = (b+c)/n
  - accuracy = 1 - error rate
  - precision (P) = a/(a+b)
  - recall (R) = a/(a+c)
  - break-even = (P+R)/2
  - F measure = 2PR/(P+R)

Reuters - Accuracy ((R+P)/2)

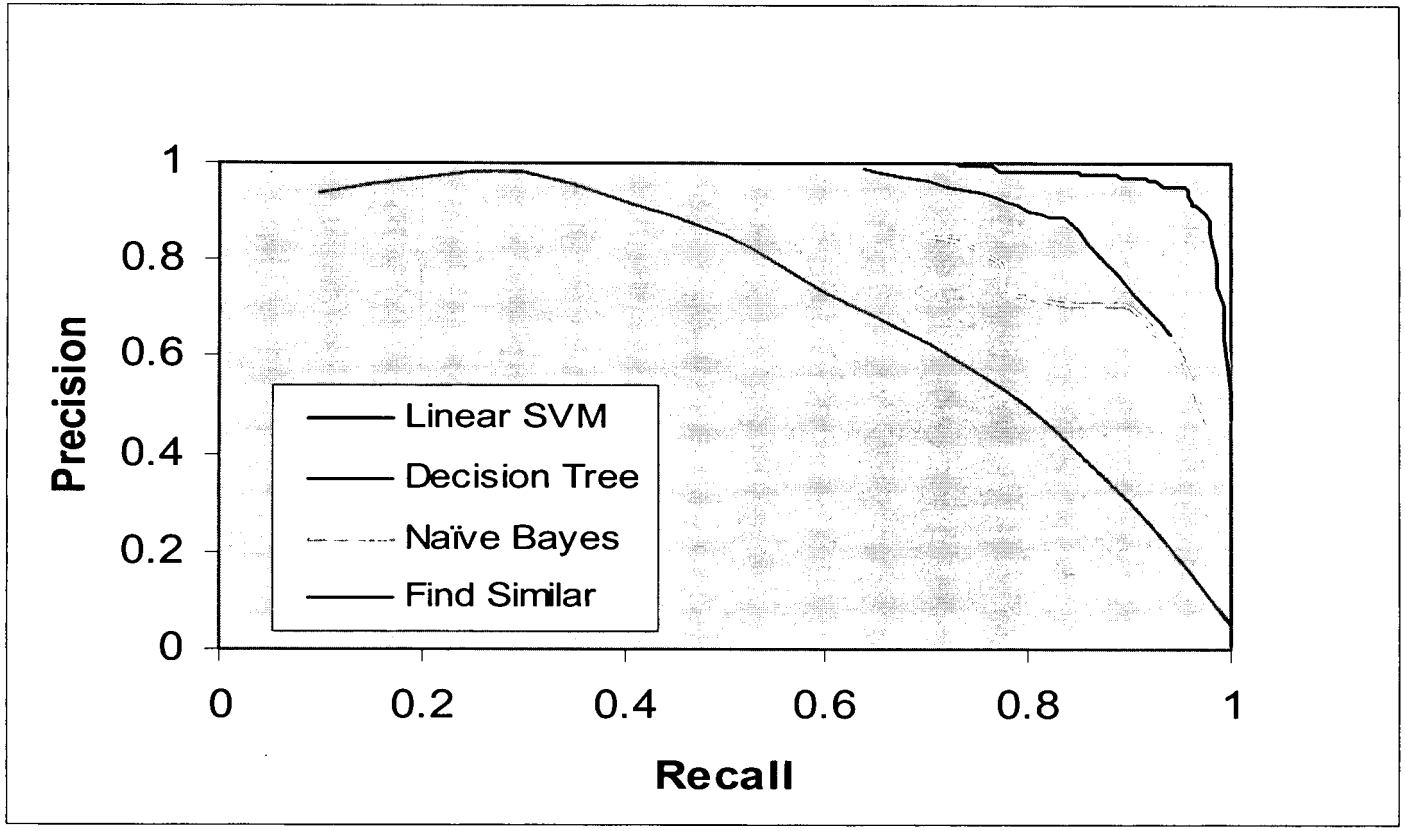
	Findsim	NBayes	BayesNets	Trees	LinearSVM
earn	92.9%	95.9%	95.8%	97.8%	98.0%
acq	64.7%	87.8%	88.3%	89.7%	93.6%
money-fx	46.7%	56.6%	58.8%	66.2%	74.5%
grain	67.5%	78.8%	81.4%	85.0%	94.6%
crude	70.1%	79.5%	79.6%	85.0%	88.9%
trade	65.1%	63.9%	69.0%	72.5%	75.9%
interest	63.4%	64.9%	71.3%	67.1%	77.7%
ship	49.2%	85.4%	84.4%	74.2%	85.6%
wheat	68.9%	69.7%	82.7%	92.5%	91.8%
corn	48.2%	65.3%	76.4%	91.8%	90.3%
Avg Top 10	64.6%	81.5%	85.0%	88.4%	92.0%
Avg All Cat	61.7%	75.2%	80.0%	N/A	87.0%

Recall: % labeled in category among those stories that are really in category

Precision: % really in category among those stories labeled in category

Break Even: (Recall + Precision) / 2

ROC for Category - Grain



Recall: % labeled in category among those stories that are really in category

Precision: % really in category among those stories labeled in category

Reuters - Sample Size (SVM)

sample set 1

category	100% samp sz	(p+r)/2	10% samp sz	(p+r)/2	5% samp sz	(p+r)/2	1% samp sz	(p+r)/2
0-acq	2876	98.3%	281	97.8%	145	97.4%	35	93.6%
1-earn	1650	97.0%	162	94.6%	80	90.4%	14	65.6%
3-money-fx	538	80.2%	55	66.3%	28	63.9%	3	41.9%
4-grain	433	95.9%	46	91.5%	21	87.0%	3	50.3%
5-crude	389	90.4%	45	82.9%	18	76.9%	3	??
6-trade	369	80.9%	40	78.2%	21	76.4%	2	12.0%
7-interest	347	79.9%	32	68.4%	17	55.3%	2	50.8%
8-ship	197	85.5%	20	57.4%	11	53.9%	2	??
9-wheat	212	92.5%	24	84.8%	11	65.7%	2	50.7%
10-corn	182	93.0%	23	78.2%	9	60.3%	1	50.9%
microtop10		93.9%		89.7%		86.0%		70.3%

sample set 2

category	100% samp sz	(p+r)/2	10% samp sz	(p+r)/2	5% samp sz	(p+r)/2	1% samp sz	(p+r)/2
0-acq	2876	98.3%	264	97.6%	139	97.3%	26	94.6%
1-earn	1650	97.0%	184	94.1%	79	90.6%	16	73.5%
3-money-fx	538	80.2%	62	72.9%	29	71.0%	5	49.9%
4-grain	433	95.9%	40	85.8%	28	88.6%	7	76.5%
5-crude	389	90.4%	35	81.6%	15	65.6%	2	??
6-trade	369	80.9%	41	80.1%	22	72.7%	5	51.6%
7-interest	347	79.9%	30	71.8%	15	63.0%	3	45.1%
8-ship	197	85.5%	21	62.8%	10	54.5%	2	50.6%
9-wheat	212	92.5%	19	80.1%	15	80.1%	5	68.3%
10-corn	182	93.0%	16	76.6%	11	69.6%	4	43.4%
microtop10		93.9%		89.6%		86.4%		75.5%

Reuters Summary

- Accurate classifiers can be learned automatically from training examples
- Linear SVMs are efficient and provide very good classification accuracy
  - Better than best previously reported results for this test collection
- Widely applicable, flexible, and adaptable representations

PFS/PSS Incident Reports

⌘ 17 largest categories (WPI - S:I:R: info)

- 3-Hardware & Device Drivers\ Device not working properly\ Modem\ Winmodem\ Aztech Sound 3-4 does not work after Windov
- 3-Hardware & Device Drivers\ Device not working properly\ Modem\ Winmodem\ USR-3COM does not work after Windows inst
- 4-Installation\ Fatal Events\ Startup Failure\ Drive not Sys'd on Install - Anti-Virus
- 4-Installation\ Pre-Install Questions\ How To: Install
- 4-Installation\ Pre-Install Questions\ How To: Install on a clean hard drive
- 4-Installation\ Add or Remove\ Reinstall\ How To: reinstall fax
- 4-Installation\ Setup Errors\ ErrMsg: "cab file corrupt" or "error in cab file"
- 6-Connectivity\ Managing Client Connections\ How To: Connect to AOL
- 6-Connectivity\ Connection Failures\ Peer to Peer networking not working after upgrade
- 6-Connectivity\ Connection Failures\ Unable to Connect to Resource\ Can Connect to AOL but Cannot Browse Internet
- 6-Connectivity\ Connection Failures\ Unable to Connect to Resource\ ErrMsg: A connection to the server could not be establish
- 6-Connectivity\ Connection Failures\ Unable to Connect to Resource\ ErrMsg: Internal Error Occurred in Windows Internet Exte
- 7-Tasks\ Startup\ Alerts and Error Messages\ Cannot Find or Initialize\ ErrMsg: Cannot find a device file needed...
- 7-Tasks\ Fatal Events\ General Protection Fault (GPF)\ ErrMsg: MSGSRV32 Caused GPF in CM8330SB.DRV (sound) on Shut
- 7-Tasks\ Fatal Events\ ErrMsg: This program has performed an illegal operation...
- 7-Tasks\ Fatal Events\ Startup Failure\ Hangs at Windows logo (splash) on startup (non-setup)
- 7-Tasks\ Fatal Events\ Hang or Lockup\ Hangs when connected to internet via DUN (Random)

PFS/PSS Incident Reports

Category	Train (695)	Test (265)
3-Hardware\Aztec	50	14
3-Hardware\USR3COM	49	25
4-Installation\Drive	36	13
4-Installation\ErrCab	33	14
4-Installation\HowToInsCleanHD	37	20
4-Installation\HowToInsFax	76	23
4-Installation\HowToInstall	48	22
6-Connectivity\AOL	55	19
6-Connectivity\ErrIntExt	33	9
6-Connectivity\ErrServer	59	30
6-Connectivity\HowToAOL	21	9
6-Connectivity\PeerToPeer	47	15
7-Tasks\ErrDevice	32	11
7-Tasks\ErrFatal	22	9
7-Tasks\ErrGPF	31	10
7-Tasks\HangsDUN	37	10
7-Tasks\HangsWindow	29	12

PFS/PSS Incident Reports

⌘Example Incident Report

S-wants help to fdisk and reformat to FAT16

I-has older applications he believes will run easier without errors on FAT 16 instead of FAT32

R-after making sure customer understands all data will be permanently lost, and consulting with mentor william co bb,
boot from the EBD, verify CD support
we proceed to use fdisk to delete the partion and recreate
formatting the C: drive now and customer wishes to discontinue call
give some basic intructions on how to run setup after formatting is done bootin g with the EBD to DOS and then how to verify prior product using win95 CD

cust has task#

⌘ Class: 4-Installation\ Pre-Install Questions\ How To: Install on a clean hard drive

PFS/PSS Example Incident

⌘Example Incident Report

Customer is having modem problems--Troubleshooting Sound 4 WinModems

S: Customer's modem does not detect after installing 98

I: Customer has installed 98 before downloading the PB updates

R: I gave him the following address and our kb address .
http://support.packardbell.com/windows98/drivers98.asp
I also gave him the q article number for this issue.

⌘ Class: 3-Hardware & Device Drivers\ Device not working properly\ Modem\ Winmodem\ Aztech Sound 3-4 does not work after Windows install

PFS/PSS Example Incident with S:I:R: coding

⌘ Example Incident Report

Customer is having modem problems--Troubleshooting Sound 4 WinModems

S: Customer's modem does not detect after installing 98

I: Customer has installed 98 before downloading the PB updates

R: I gave him the following address and our kb address .
http://support.packardbell.com/windows98/drivers98.asp
I also gave him the q article number for this issue.

⌘ S:I:R: Coding

N_Customer N_is N_having N_modem N_problems--N_Troubleshooting N_Sound 4 N_WinModems

S_Customer's S_modem S_does S_not S_detect S_after S_installing S_98

I_Customer I_has I_installed I_98 I_before I_downloading I_the I_PB I_updates

R_I R_gave R_him R_the R_following R_address R_and R_our R_kb R_address
R_http://support.packardbell.com/windows98/drivers98.asp
R_I R_also R_gave R_him R_the R_q R_article R_number R_for R_his R_ssue.

PFS/PSS Incident Reports

- Notes about text
  - Many typos, abbreviations, acronyms -- did nothing about cleaning this up, normalizing
  - Used only individual words -- no phrases (e.g., “Windows NT” == “Windows”, “NT”)
  - Heuristics to find S: I: R: fields

PFS/PSS - Timing Results

- Training Time - learning models
  - 17 categories; 695 examples
  - 10 secs wall time (266MHz, NTServer 4.0)
- Classification Time
  - 17 categories; 265 examples
  - 1 sec wall time (266MHz, NTServer 4.0)

Category: Task\HangDUN

- 7-Tasks\ Fatal Events\ Hang or Lockup\ Hangs when connected to internet via DUN (Random)

Example SVM features (plain) - $\vec{w}$

- 603579 hangs
- 482204 locks
- 393038 temporary
- 391808 freezes
- 335581 freezing
- 330979 internet
- 282411 winsocks
- 264000 group
- -233082 setting
- -218972 install
- -203328 connectoid
- -195753 icon
- -195744 setup
- -189621 connection
- -185406 command
- -175512 close

Category: Task\HangDUN

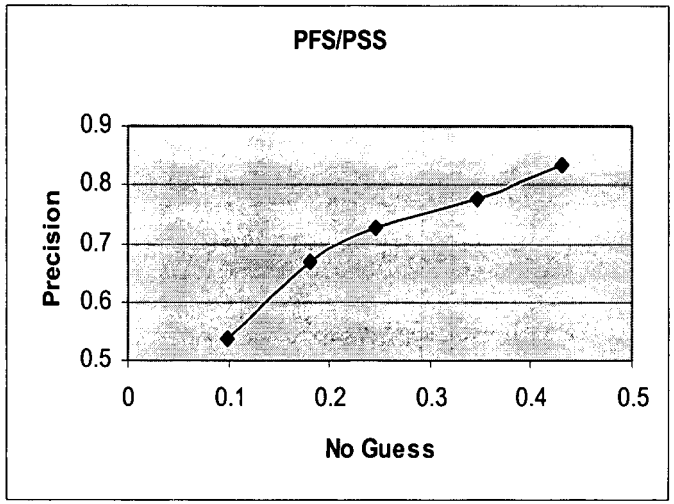
- 7-Tasks\ Fatal Events\ Hang or Lockup\ Hangs when connected to internet via DUN (Random)

Example SVM features (split) - $\vec{w}$

- 305470 n_when
- 291103 r_temporary
- 291103 temporary
- 285835 hangs
- 266035 s_hangs
- 265592 freezing
- 265592 s_freezing
- 259604 s_up
- -228544 file
- -209362 s_not
- -207037 win98
- -197706 r_into
- -189775 setup
- -167039 r_win98
- -154349 message
- -149015 n_upgrade

PFS/PSS - Accuracy Results

- Overall Accuracy with default threshold
  - Precision: 84%
  - No Guess: 43%
- Vary Threshold to decrease no guess %



PFS/PSS - Vary Threshold

VARY p -- 300p	n test	nguess	ncorr	prec	recall	no guess
p=.1	265	373	201	0.539	0.758	0.098
p=.2	265	270	181	0.67	0.683	0.181
p=.3	265	230	167	0.726	0.63	0.245
p=.4	265	193	150	0.777	0.567	0.347
p=.5	265	159	133	0.836	0.502	0.43

PFS/PSS - Example "Miss"

- Incident text

  N: Uninstalled 98 and remarked out 3.1 now wants to rerun setup

  S: Abel is on the he has been transferred from the 3.1 queue, he wants to install win98, he had previous issues and had to uninstall and go back to 3.1 queue and be retransferred to 98 queue.

  R: Cust had some utilities loading in autoexec that they remarked out in 3.1 and now is attempting to reinstall win98, Would like to have some moral support while he goes through the process as he had issues previously on boot up and never really made it into the windows 98 program. he had been on the phone for about 2 hours with 98 and 3.1 support, so after the copy file progress began, I let him go eat lunch and informed to call back with the task number

  should he encounter any issues again. Closed not resolved

- Category:

  4-Installation\ Pre-Install Questions\ How To: Install

PFS/PSS - Example "Miss"

- Incident text

  N: Legacy Sound Card Drivers Were Not Removed When Win98 Reinstalled

  S: 1) installed Win98 upgraded motherboard & CPU then reinstalled win98 2) at boot MSGSRV32 th is program has performed an illegal operation ... MSGSRV32 caused a segment not present fault in module unknown

  I: 1) Legacy Sound Card Drivers Were Not Removed When Win98 Reinstalled

  R: 1) boot safe mode 2) selective sta rtup unchecked all five boxes 3) boot normal mode win98 setup continued 4) selective startup process system.ini 5) issue is in [drivers] wave 1 aux 1 both pointed to old sound card on old motherboard rem both 6) restored normal startup 7) boot normal mode page fault in ECIDM 8) found in startup folder mentor Jeffri Sander rem runkey 9) boot normal mode 10) all issues resolved 11) closed

- Category:

  7-Tasks\ Fatal Events\ ErrMsg: This program has performed an illegal operation...

PFS/PSS - Confusion Matrix (p=0.5)

CONFUSION MATRIX - 300p																		
	3-Hardw	3-Hardw	4-Install	4-Install	4-Install	4-Install	4-Install	6-Conne	6-Conne	6-Conne	6-Conne	6-Conne	7-Tasks	7-Tasks	7-Tasks	7-Tasks	7-Tasks	Punt
3-Hardware\Aztec	11	1																2
3-Hardware\USR3COM	2	11				1												11
4-Installation\Drive			8														1	4
4-Installation\ErrCab				11	1		1											3
4-Installation\HowToInsCleanHD					5		3											12
4-Installation\HowToInsFax						19												4
4-Installation\HowToInstall			1	1			4									1		16
6-Connectivity\AOL								4		3						1		12
6-Connectivity\ErrIntExt									7	4								1
6-Connectivity\ErrServer									1	13								17
6-Connectivity\HowToAOL								2			1							6
6-Connectivity\PeerToPeer							1					11				1		3
7-Tasks\ErrDevice				1									7					3
7-Tasks\ErrFatal														2				7
7-Tasks\ErrGPF		1													8			2
7-Tasks\HangsDUN		1														3		7
7-Tasks\HangsWindow																	8	4

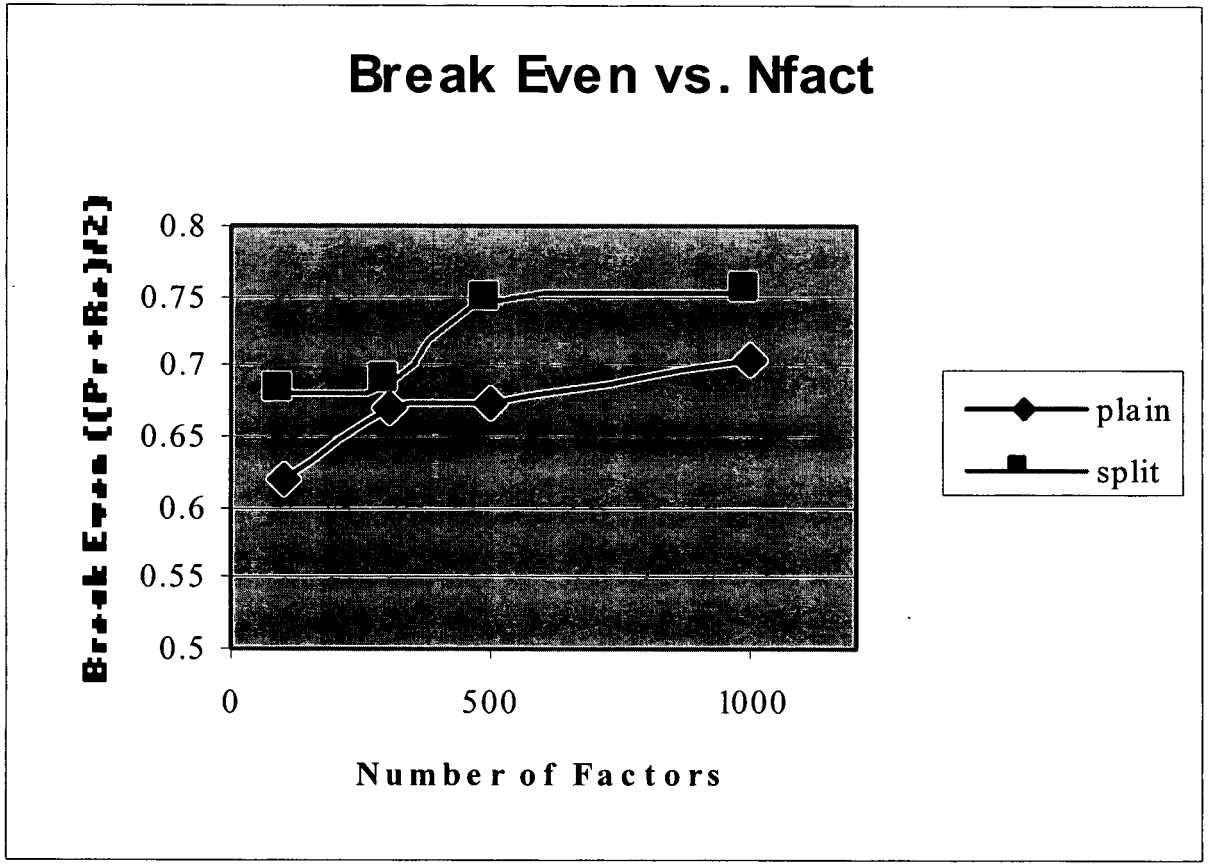
PFS/PSS - Confusion Matrix (p=0.1)

CONFUSION MATRIX - 300p																			
	3-Hardw	3-Hardw	4-Install	4-Install	4-Install	4-Install	4-Install	6-Conne	6-Conne	6-Conne	6-Conne	6-Conne	7-Tasks	7-Tasks	7-Tasks	7-Tasks	7-Tasks	Punt	Ntest
3-Hardware\Aztec	14	8					2	1		1								0	14
3-Hardware\USR3COM	3	16	1	1				1	1	2		1			1		1	3	25
4-Installation\Drive			9		2	1	2						1				2	1	13
4-Installation\ErrCab			1	13			1						2					0	14
4-Installation\HowToInsCleanHD			1	4	15		5						1	1			1	1	20
4-Installation\HowToInsFax		1	1		7	23	2									1		0	23
4-Installation\HowToInstall			1	2	7	1	8			2		1	1	2	3		1	4	22
6-Connectivity\AOL		1	1			5	3	15	2	4	7	2	4		2	4	1	3	19
6-Connectivity\ErrIntExt								1	9	4								0	9
6-Connectivity\ErrServer								3	2	18	1					2	1	7	30
6-Connectivity\HowToAOL	2							1			7							1	9
6-Connectivity\PeerToPeer												13						2	15
7-Tasks\ErrDevice			2	2			1			2		1	7					1	11
7-Tasks\ErrFatal														9				0	9
7-Tasks\ErrGPF														2	10			0	10
7-Tasks\HangsDUN								1		1		1				6	3	0	10
7-Tasks\HangsWindow		2	2											0			11	1	12
																			265

PFS/PSS Plain vs. Split

- Plain
  - Precision: 84%
  - No Guess: 43%
- Split by S: I: R: (using heuristics)
  - Precision: 81%
  - No Guess: 37%

Performance vs. Nfact (Plain vs. SIR Split)



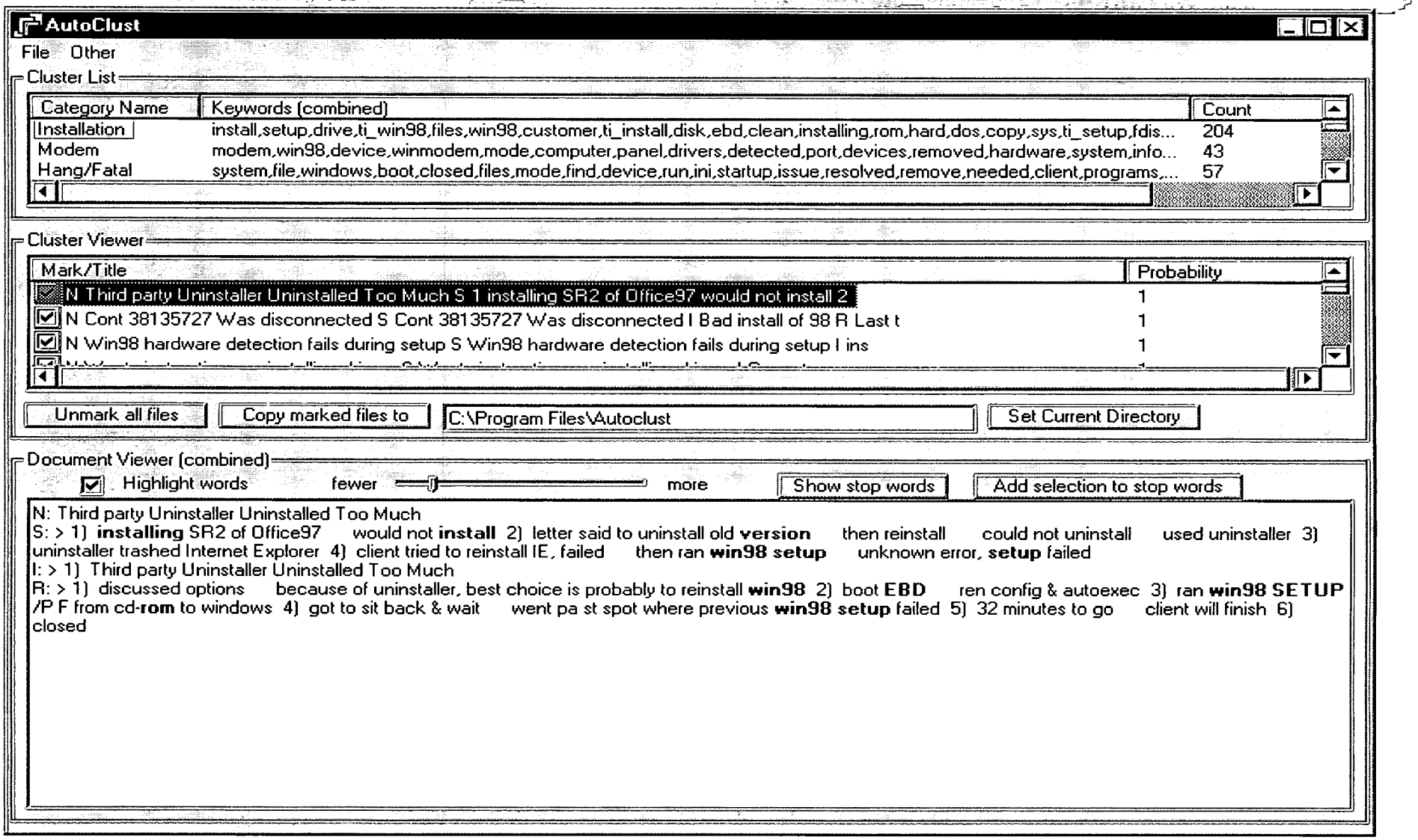
PFS/PSS Summary

- Very encouraging results
- Reasonably accurate classifiers can be learned automatically from training examples
- Methods for improving
  - Hierarchical classification
  - Vocabulary normalization
  - Text cleaning (spelling correction; s:i:r: recognition)
  - Phrases

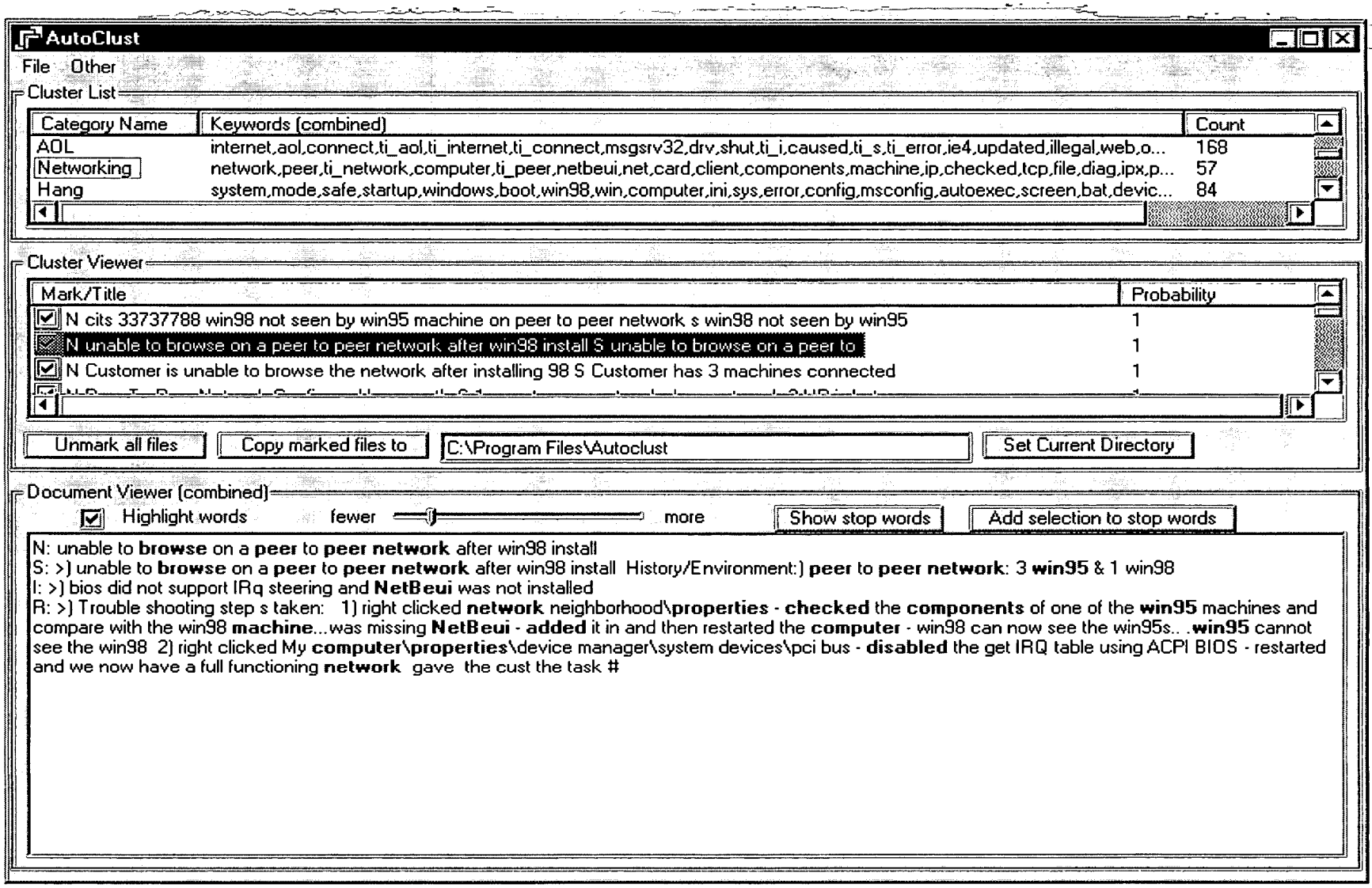
More Information

- html View, Organized by Category
  - \\sdumais\public\pfs-pss\wpi*.htm
- General Info on Classification
  - http://research.microsoft.com/~sdumais
- Code for Classification and Clustering
  - http://msrweb/dtas/software&tools.htm

PFS/PSS Clustering



PFS/PSS Clustering

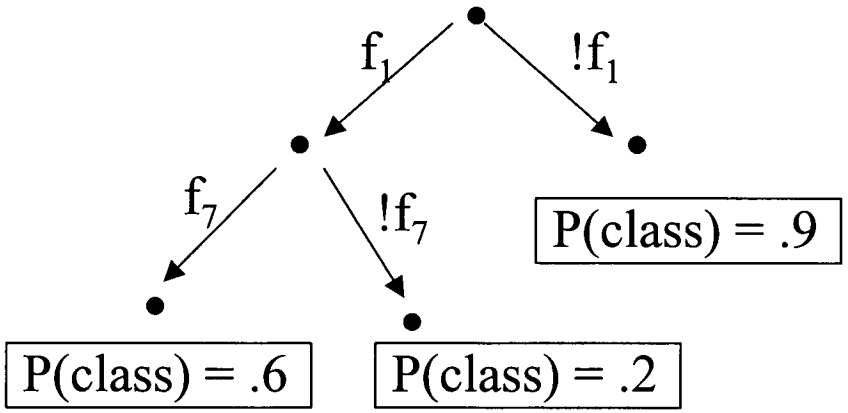


Find Similar

- Aka, relevance feedback
- Rocchio $$w_j = \beta \sum_{i \in rel} \frac{x_{i,j}}{n} - \gamma \sum_{i \in non\_rel} \frac{x_{i,j}}{N-n}$$
- Classifier parameters are a weighted combination of weights in positive and negative examples -- "centroid"
- New items classified using: $\sum_j w_j \cdot x_j$
- Use all features, idf weights, $\gamma = 0$

Decision Trees

- Learn a sequence of tests on features, typically using top-down, greedy search
- Binary (yes/no) or continuous decisions

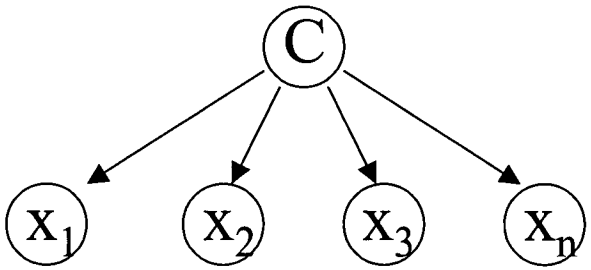


Naïve Bayes

- Aka, binary independence model
- Maximize: Pr (Class | Features)

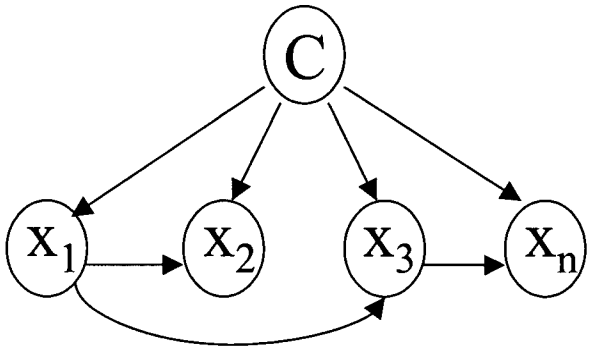
$$P(class \mid \vec{x}) = \frac{P(\vec{x} \mid class) \cdot P(class)}{P(\vec{x})}$$

- Assume features are conditionally independent
  - math easy; surprisingly effective

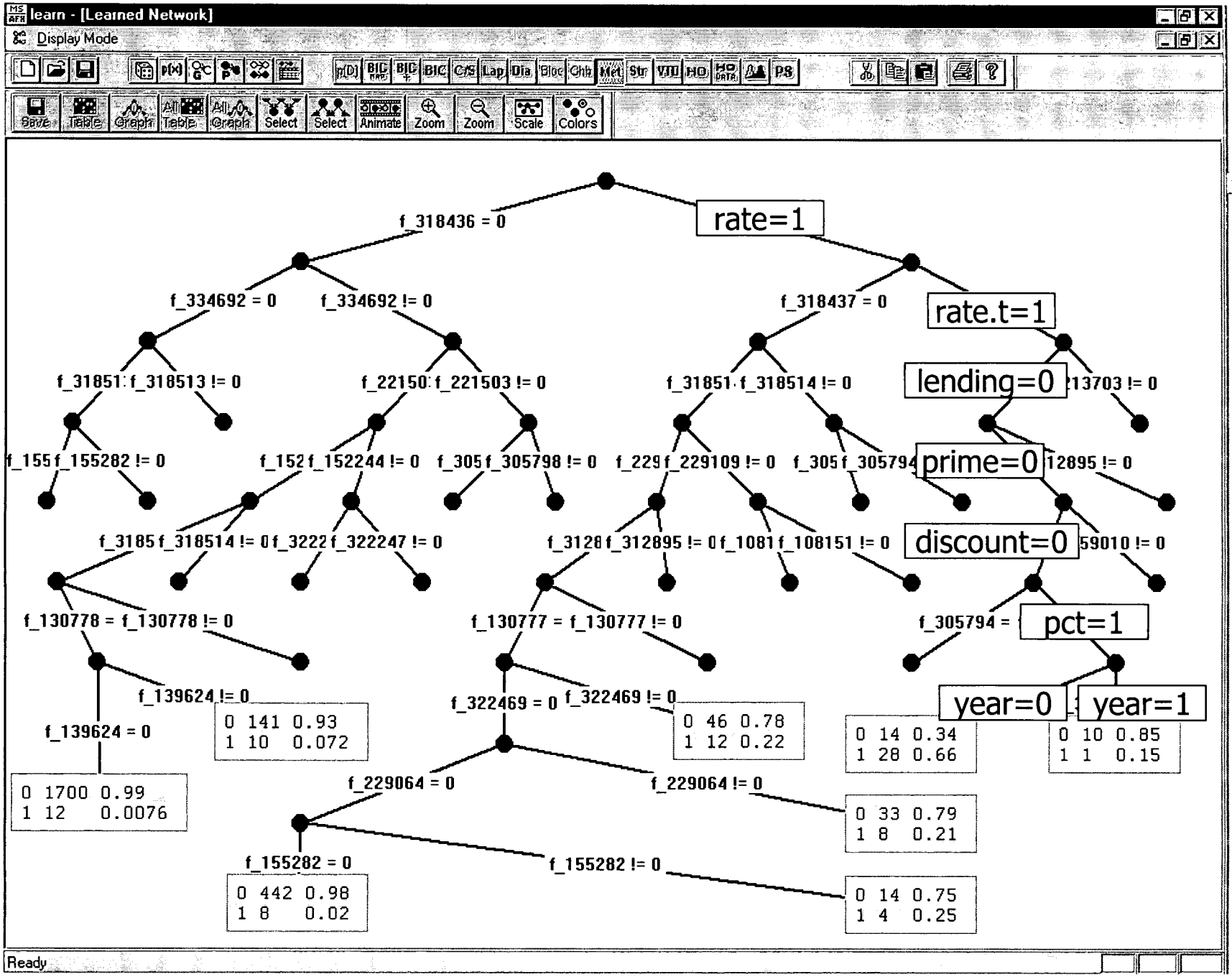


Bayes Nets

- Maximize: Pr (Class | Features)
- Does not assume independence of features - dependency modeling



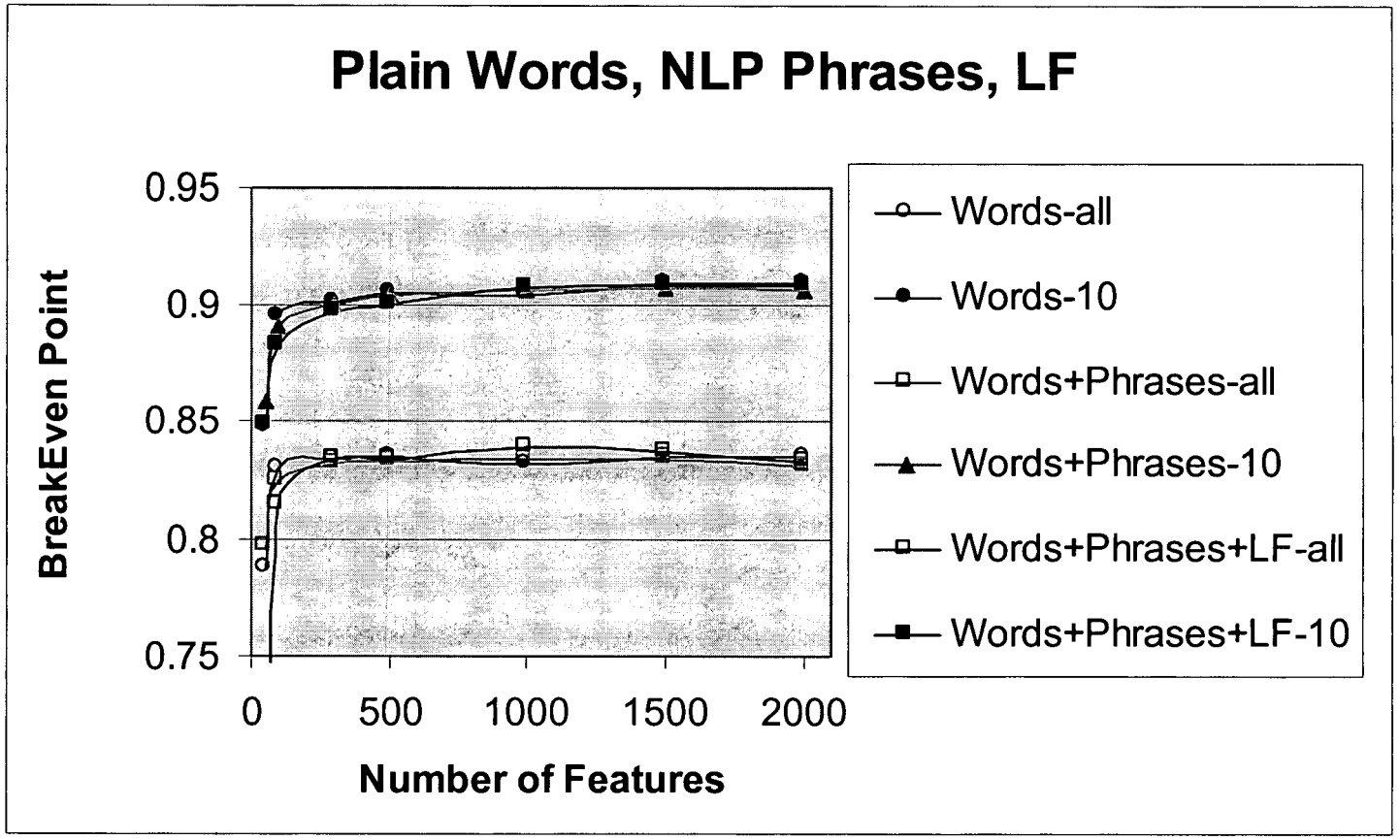
Category: “interest”



Reuters - Other Experiments

- Simple words vs. NLP-derived phrases
  - NLP-derived phrases
    - factoids (April_8, Salomon_Brothers_International)
    - mulit-word dictionary entries (New_York, interest_rate)
    - noun phrases (first_quarter, modest_growth)
  - No advantage for Find Similar, Naïve Bayes, SVM
  - Vary number of features
- Binary vs. 0/1/2 features
  - No advantage of 0/1/2 for Decision Trees
  - Need to try w/ SVM

Number of Features - NLP



Text Classification Horizon

- Other applications
  - OHSUMED, TREC, spam vs. not-spam, Web
- Text representation enhancements for SVM model
- Use of hierarchical category structure
- UI for semi-automatic classification
- Dynamic interests

Exhibit F

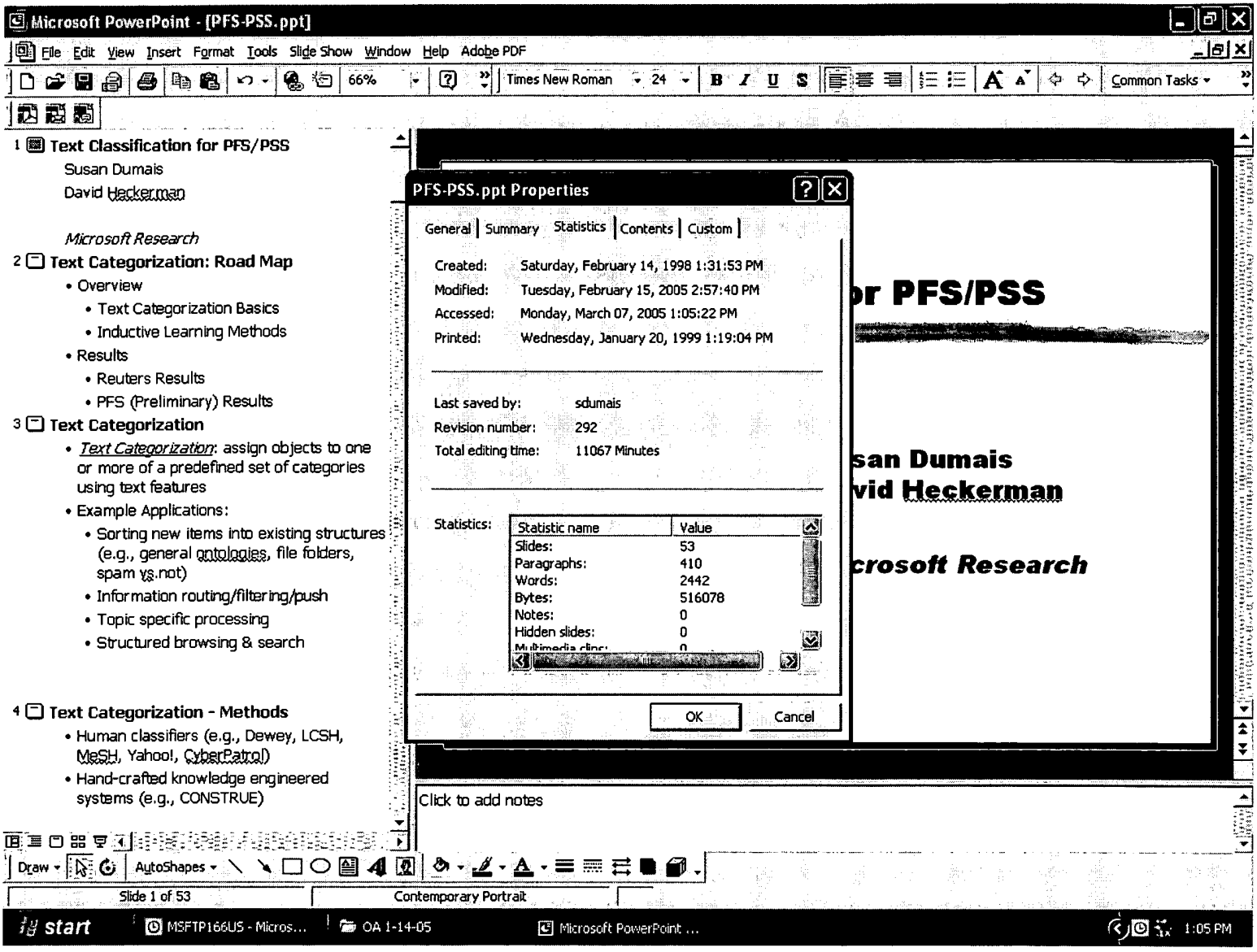

Microsoft PowerPoint - [PFS-PSS.ppt]
File Edit View Insert Format Tools Slide Show Window Help Adobe PDF
Times New Roman 24 Common Tasks
1 Text Classification for PFS/PSS
Susan Dumais
David Heckerman
Microsoft Research
2 Text Categorization: Road Map
• Overview
• Text Categorization Basics
• Inductive Learning Methods
• Results
• Reuters Results
• PFS (Preliminary) Results
3 Text Categorization
• Text Categorization: assign objects to one or more of a predefined set of categories using text features
• Example Applications:
• Sorting new items into existing structures (e.g., general ontologies, file folders, spam vs.not)
• Information routing/filtering/push
• Topic specific processing
• Structured browsing & search
4 Text Categorization - Methods
• Human classifiers (e.g., Dewey, LCSH, MeSH, Yahoo!, CyberPatrol)
• Hand-crafted knowledge engineered systems (e.g., CONSTRUE)
PFS-PSS.ppt Properties
General Summary Statistics Contents Custom
Created: Saturday, February 14, 1998 1:31:53 PM
Modified: Tuesday, February 15, 2005 2:57:40 PM
Accessed: Monday, March 07, 2005 1:05:22 PM
Printed: Wednesday, January 20, 1999 1:19:04 PM
Last saved by: sdumais
Revision number: 292
Total editing time: 11067 Minutes
Statistics:
Statistic name Value
Slides: 53
Paragraphs: 410
Words: 2442
Bytes: 516078
Notes: 0
Hidden slides: 0
OK Cancel
or PFS/PSS
san Dumais
vid Heckerman
crosoft Research
Click to add notes
Draw AutoShapes
Slide 1 of 53
Contemporary Portrait
start
MSFTP166US - Micros...
OA 1-14-05
Microsoft PowerPoint ...
1:05 PM

EXHIBIT G

Bringing Order to the Web:
Automatically Categorizing Search Results

Susan Dumais, MSR

Hao Chen, Intern UCBerkeley

Overview

- Organize search results … using classification
- UI and user study
- Possibilities for MSN

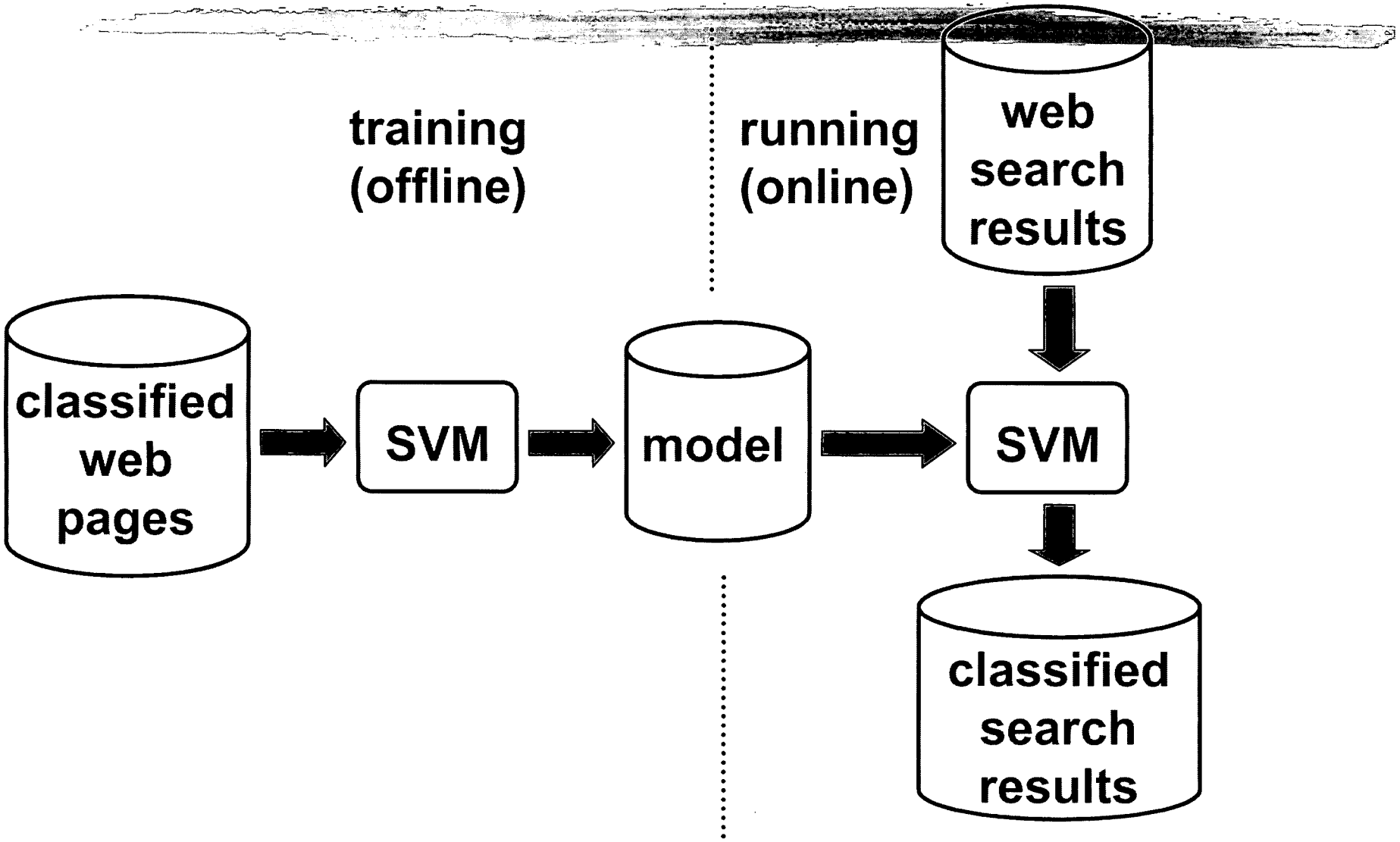
Organizing Search Results

- Query -> Ranked list
- Beyond the ranked list
  - Clustering
    - e.g., Zamir, Maarek, Hatana, fDTAS/MSN
    - but, slowish, and difficulties in labeling categories
  - Classification
    - e.g., Northern Light (for known information), Inktomi?, Chakrabarti, Mladenic
    - easily understood category labels

Classifying Search Results

- Combines the advantages of
  - Broad coverage from search engines
  - Manually compiled directory structure
- Two main system components
  - *Classification models*:
    - Trained on manually classified pages (offline)
    - Classify search results on-the-fly (online)
  - *UI for integrating search and structure*:
    - Present results in a useful and usable form

System Components



LookSmart Directory

- 450k Web pages; 17k categories
- Expts: 13 Top-Level; 150 Second-Level
- Top-level categories

0.0 (4982) Automotive
0.1 (31599) Business & Finance
0.2 (46000) Computers & Internet
0.3 (88697) Entertainment & Media
0.4 (25370) Health & Fitness
0.5 (22959) Hobbies & Interests
0.6 (11484) Home & Family
0.7 (35157) People & Chat
0.8 (58002) Reference & Education
0.9 (19667) Shopping & Services
0.10 (38968) Society & Politics
0.11 (23559) Sports & Recreation
0.12 (43682) Travel & Vacations

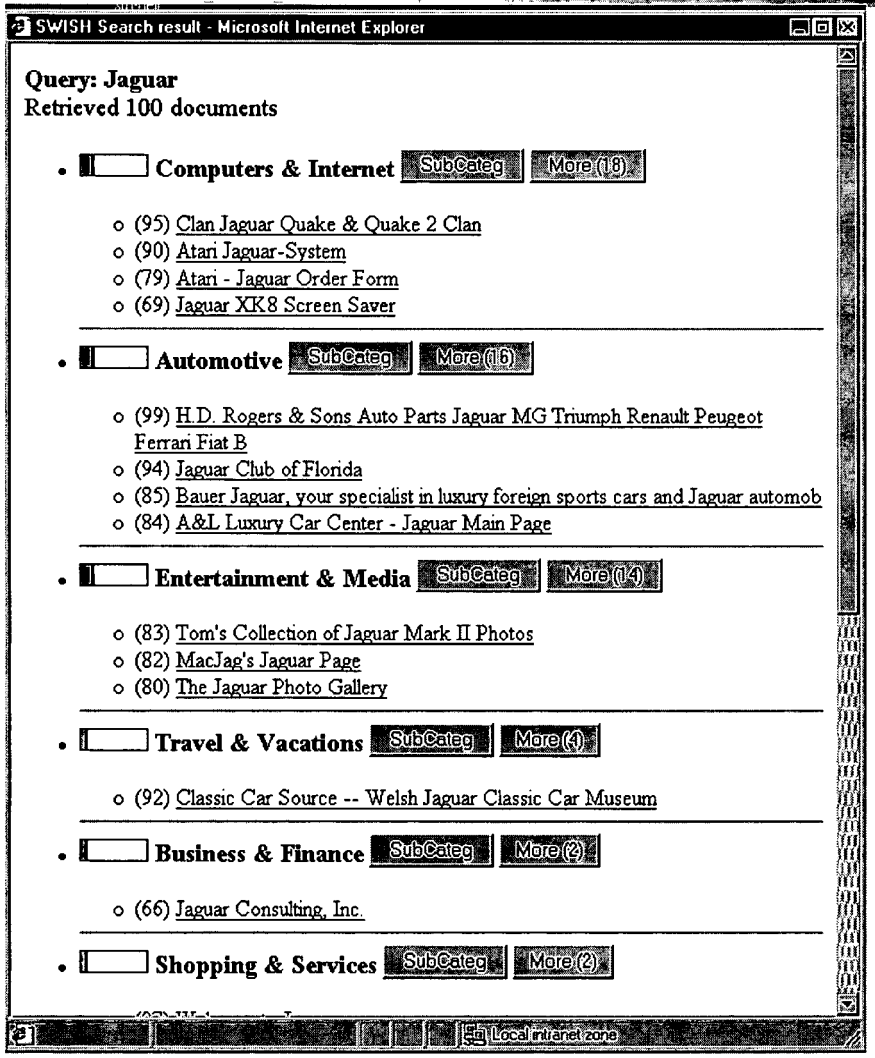
Training Classifiers

- Support Vector Machine (SVM)
  - Category models learned from examples
  - Model = weighted vector of words
  - Accurate and efficient
- Binary classification
  - Learn one model per category (in-vs-out)
  - Each document belongs to one or more categories

Training Classifiers

- Experiments to build accurate classifiers
  - Model: SVM vs LDA ... SVM slightly better
  - Number of features/category (500-1500) ... small diffs
  - Other parametric variations (c, p) ... small diffs
  - Length of document (full vs. summary)
    - Train full - Test full — 71% avg break-even
    - Train full - Test sum — 51% avg break-even
    - Train sum - Test sum — 61% avg break-even
  - Hierarchical vs. flat category structure
  - Binary vs. rate-based representation

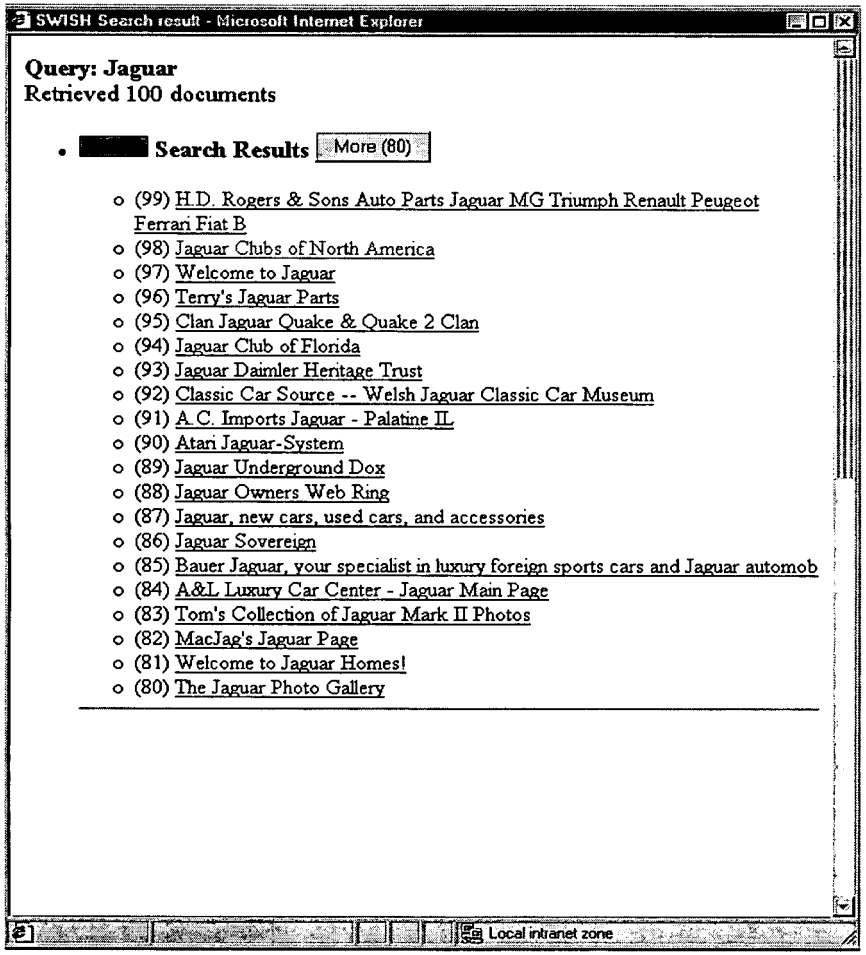
The Category Interface

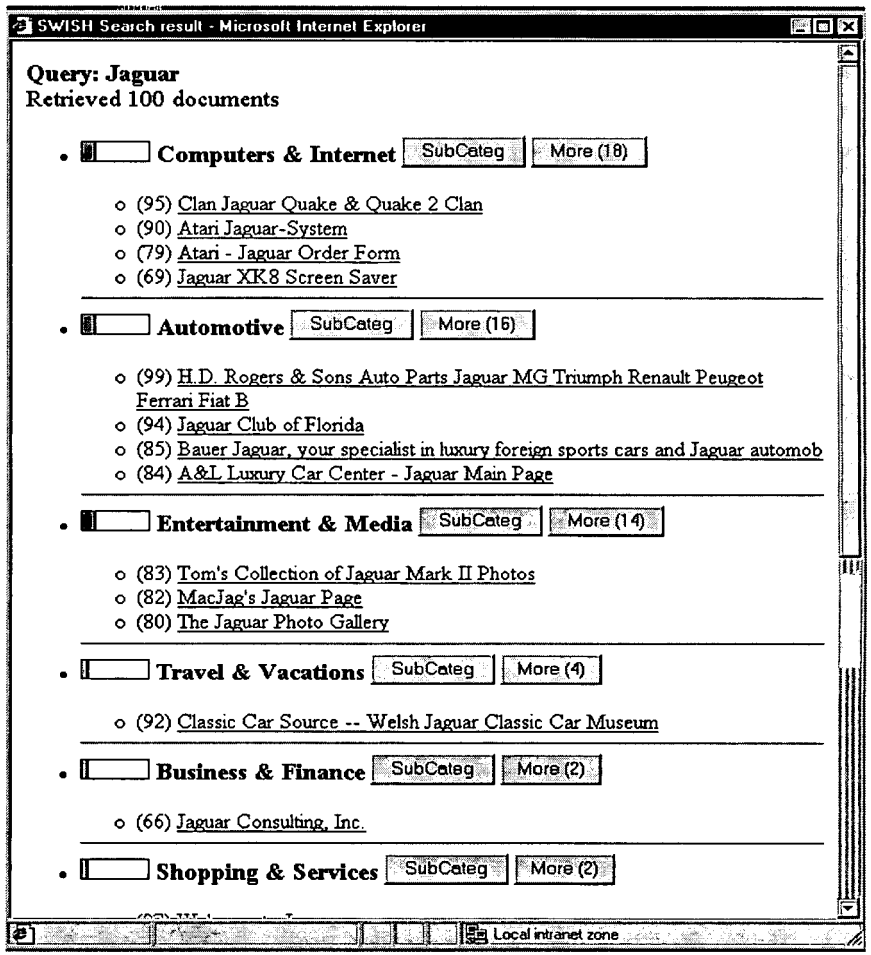


Interface Characteristics

- Operations on Categories
  - Show More
  - Sub Categorize
- Information Overlays
  - Hover text to show summary & parent-child
- Distilled Information Display
  - How to rank categories; how many to present?
  - How to rank docs within categories; how many to present?

User Study
List vs. Category Interface

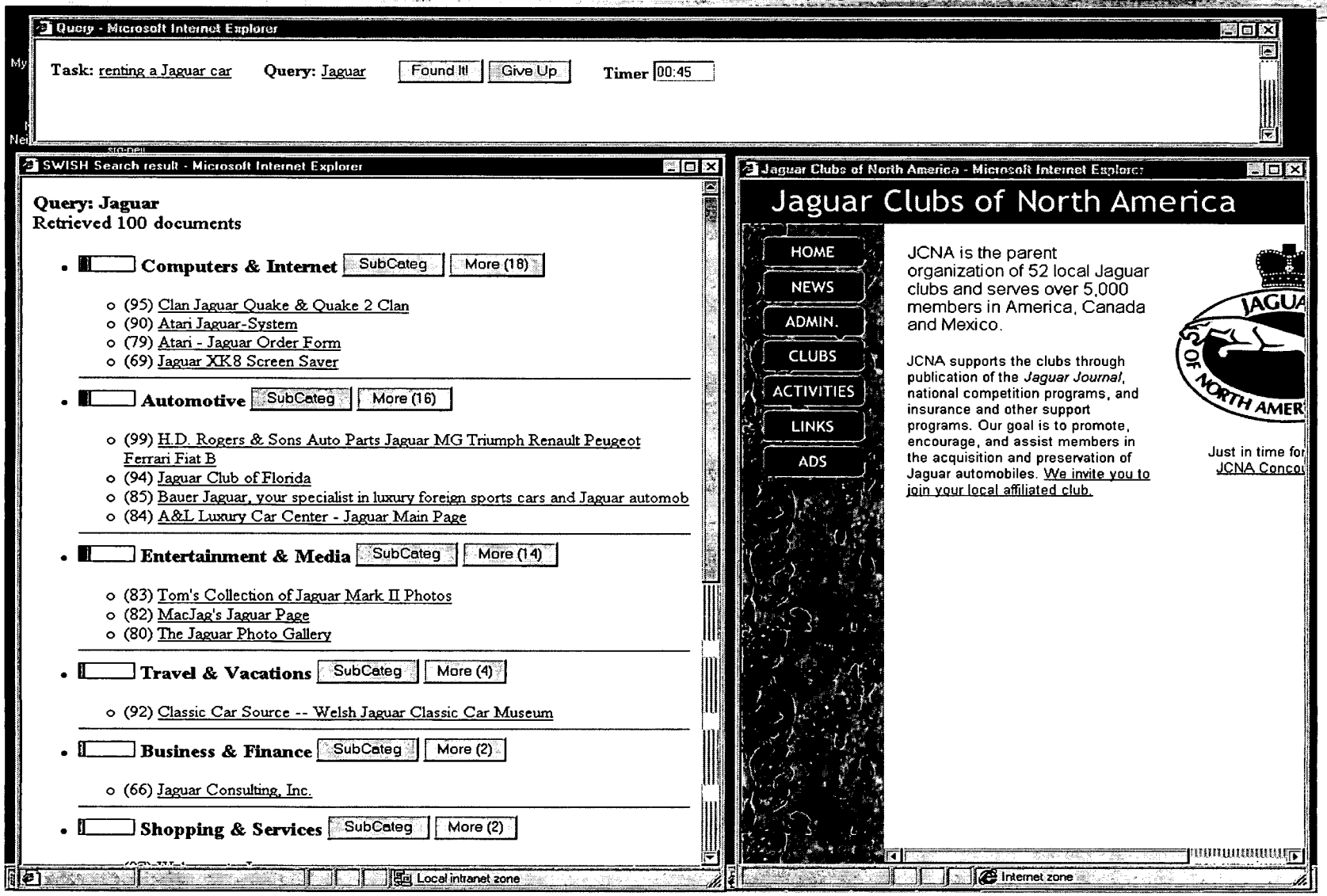




User Study

- Participants:
  - 18 intermediate Web/IE
- Search Tasks:
  - 15 tasks w/ each interface
  - e.g., *"Find home page for Seattle Art Museum"*
  - we issued queries and cached results - top 100 from (old) MSNWebSearch
  - takes about 2 hours

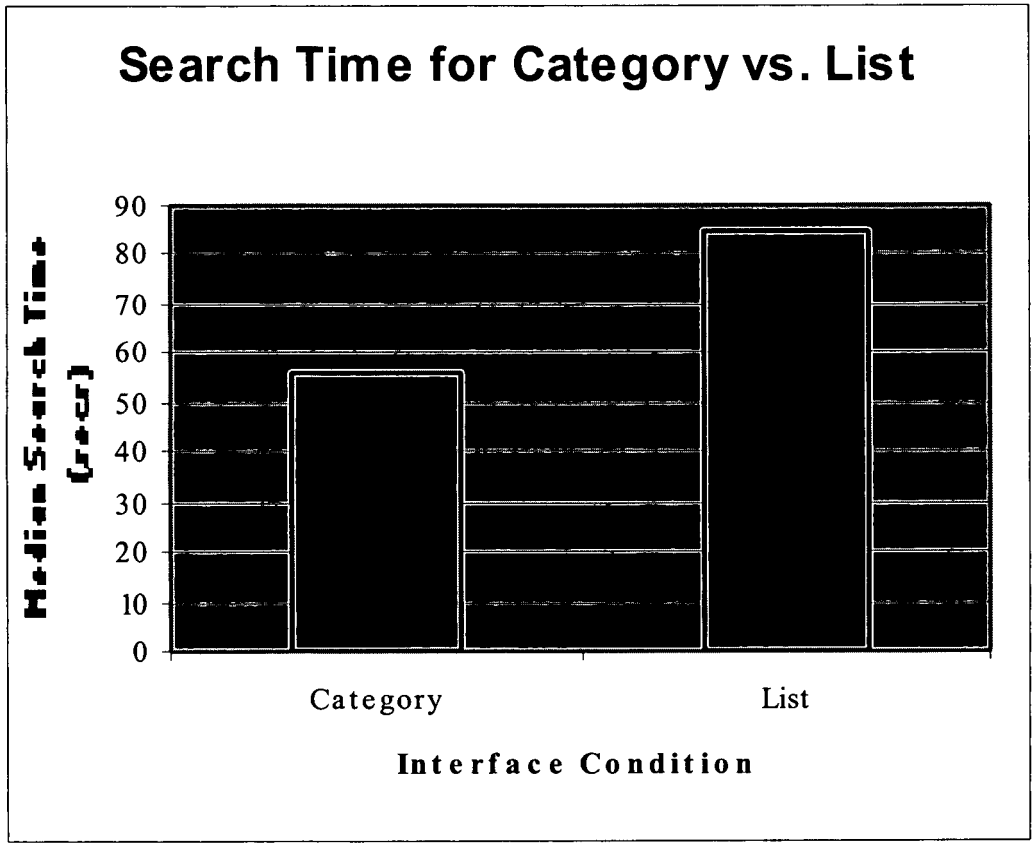
User Study Screen



Subjective Results

- Category interface vs. List interface
  - *Liked using it* (6.7 vs. 4.3)
  - *Easy to use* (6.4 vs. 3.9)
  - *Confident that I could find information if it was there* (6.3 vs. 4.4)
  - *Easy to get a good sense of the range of alternatives* (6.4 vs. 4.2)
  - *Prefer this to my usual search engine* (6.4 vs. 4.3)
- No reliable difference about usefulness of interface features (hover text, expansion)

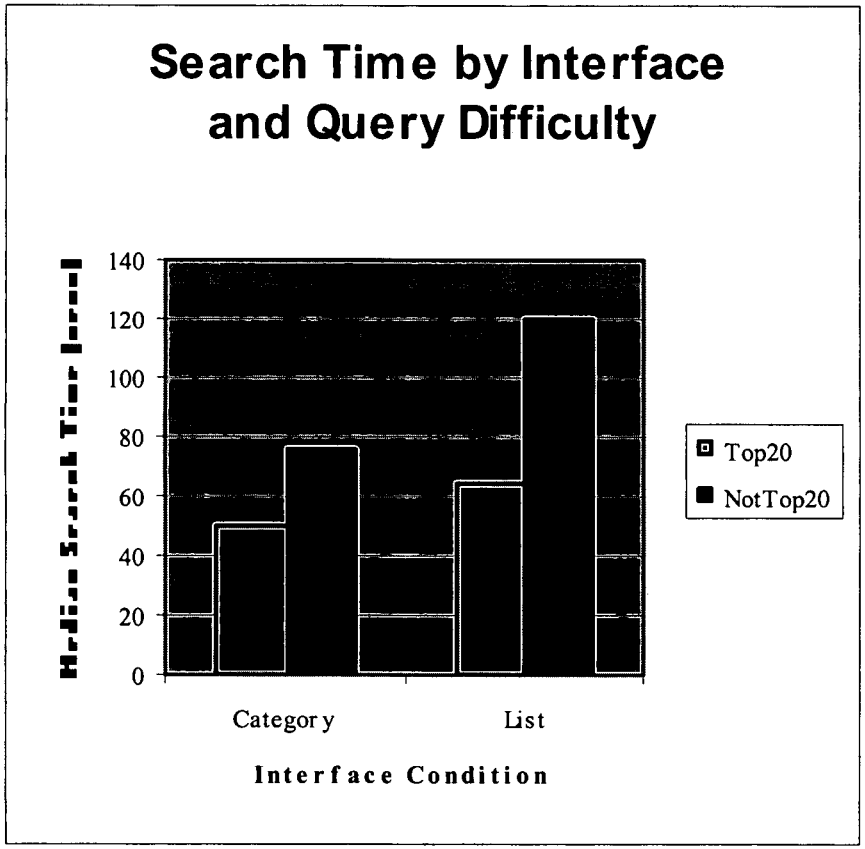
Search Time



Category: 56 secs
List: 85 secs

==> 50% faster w/ Category interface

Search Time by Query Difficulty



Top20: 57 secs

NotTop20: 98 secs

Category faster for both

Use of Interface Features

- Hover text
  - Category 2.99 List 4.60
- Show Page
  - Category 1.29 List 1.41
- ShowMore and SubCateg
  - Category 0.78 List 0.48

Variability

- Across subjects
  - 37 secs/task -> 142 secs/task
- Across queries
  - category: 24 secs -> 158 secs
  - list: 18 secs -> 217 secs
  - fastest - *"Ford's theater"*
  - slowest - *"Books about the author, Dylan Thomas"*

Summary

- Organizing Results
  - Category models trained on LookSmart
  - Allows classification of new docs on-the-fly
- User Interface
  - Organize documents by category structure
  - Operations on the category structure
  - Interaction style, distilled information display
- User Study
  - Demonstrates advantages of category interface

Issues

- Matching function
  - Query_Match: rank of page, and sometimes match score
    Categ_Match: p(category for each page)
- UI developments
  - Heuristics for automatic expansion of hierarchy
  - Integration with additional matching information
- End-to-end user study
- Speed
  - Especially, getting results from search engines

Possibilities for MSN?

- Classification
  - Organize time-critical info like news stories or breaking events that aren't in LS directory
  - Multiple directories
  - Support editorial process
- User Interface
  - Tight coupling of structure and search
  - Information overlays
- Others?

Live Demo

⌘...

MSN Search Result for - Jaguar - Microsoft Internet Explorer provided by ITG

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Print Edit

Address http://search.msn.com/results.asp?RS=CHECKED&UN=doc&CO=20&FORM=MSNH&TK=2&MT=Jaguar&zip=98033 Go

msn Microsoft

Home | Hotmail | People&Chat | Shopping | MSN Search

Search Home | Saved Results | Feedback | Help

Search the Web for:

Jaguar Search Use AltaVista Advanced Search

Win $5,000! Yellow Pages New Search

Results: containing '**Jaguar**' next >>

Featured Sites about

- **Jaguar**RN - Go directly to the Jaguar site.
- **Get Jags Tickets** - Order form for Jags season tickets.
- **Watch a Game on TV** - 1999 NFL national television schedule.
- **Get a Jags Update** - Jaguars news from MSNBC Sports.

Web Directory Topics

- Makes, Models & Clubs : European Makes : **Jaguar**, Jensen, Mercedes-Benz, Lamborghini
- Auto : Maintenance & Repair : **Jaguar**, 4WDs, BMW, Chevrolet
- Mammals : Wildcats : **Jaguars**, Guides & Directories, Bobcats, Cheetahs
- Football : AFC Central Teams : **Jacksonville Jaguars**, Baltimore Ravens, Cincinnati Bengals, Cleveland Browns

Web Directory Sites 1 - 20 of 259

The Jaguar Collection
http://www.collection.co.uk
The source for Jaguar key chains, desk accessories, personal fragrance and other official merchandise and gift items.

Jaguar 98 Factsheet - AZA Species Survival Plan
http://aza.org/Programs/SSP/ssp.cfm?ssp=46
Offers a physical description, provides a habitat range, describes the social organization, and notes the conservation status.

Jaguar - The Wild Ones
http://www.thewildones.org/Belize/jaguar.html
Find out the other names that this large, spotted cat goes by. Listen to Jaguar sound clips.

NFL.com - Jacksonville Jaguars
http://www.nfl.com/jaguars/
Game previews, schedule, team roster and draft information on one of the league's newest

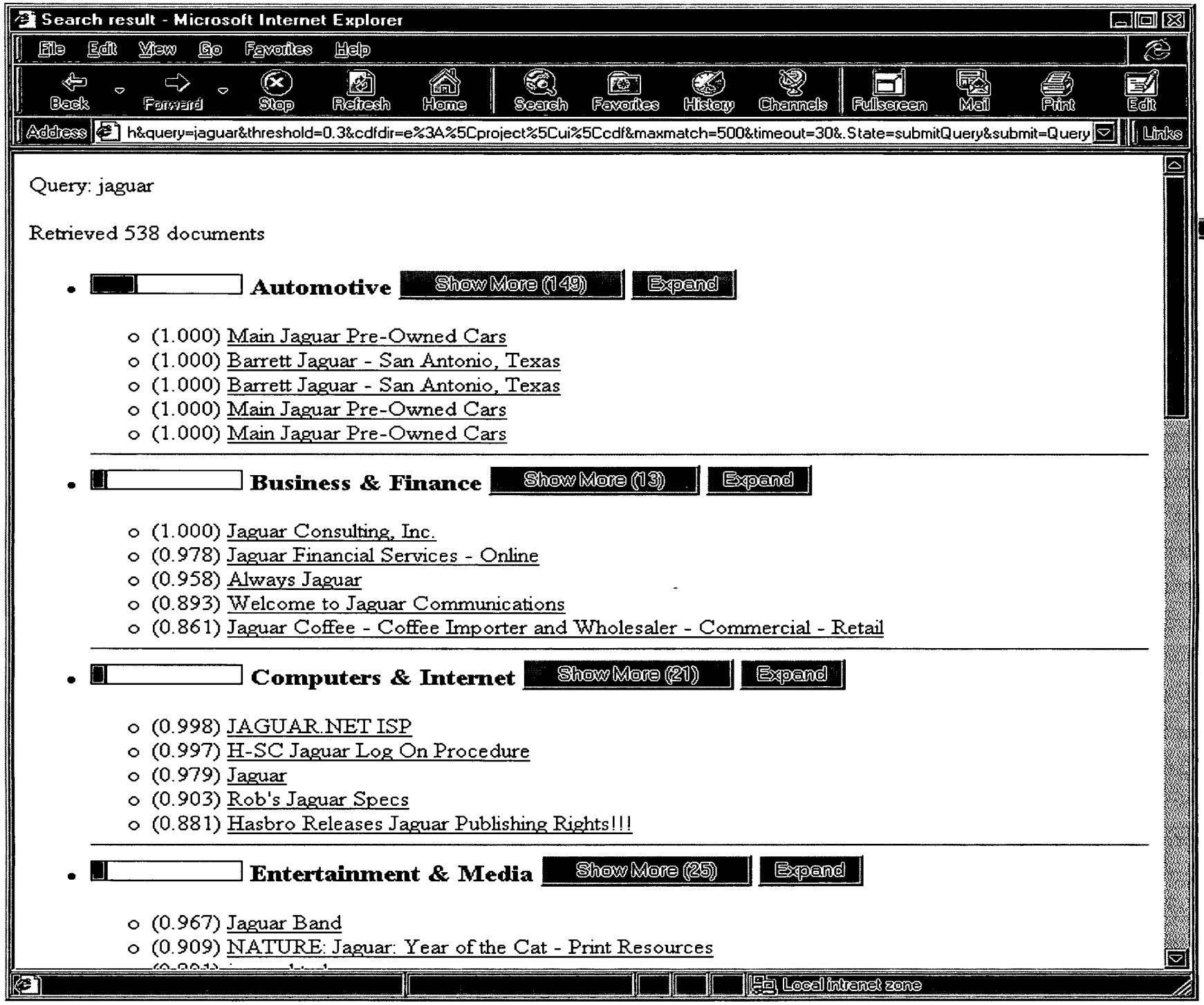
http://ads.msn.com/ads/redirect.dll/CID=000512338d2c1a4400000000/AREA=IMGSRH?image=http://ads Internet

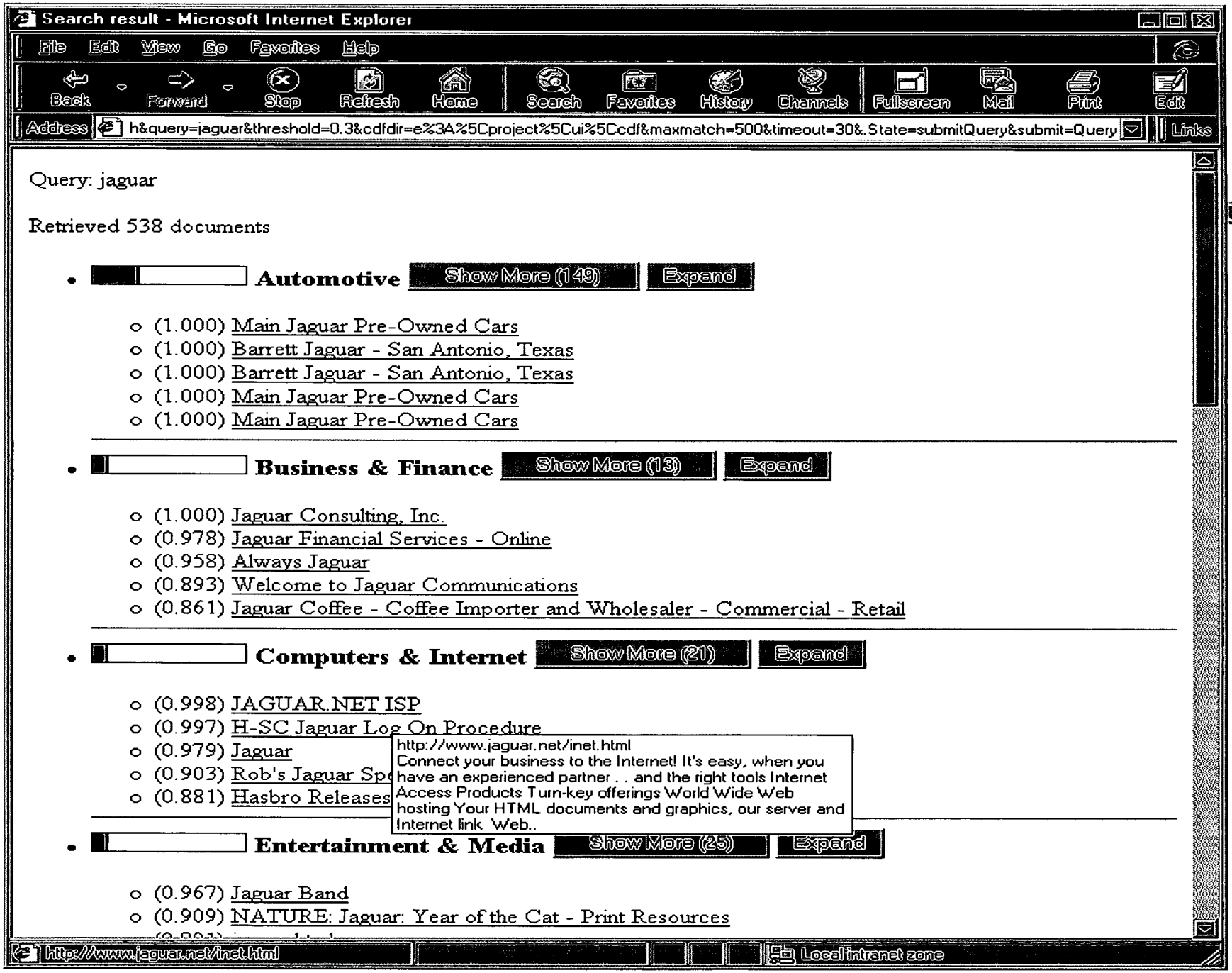
Gave Up

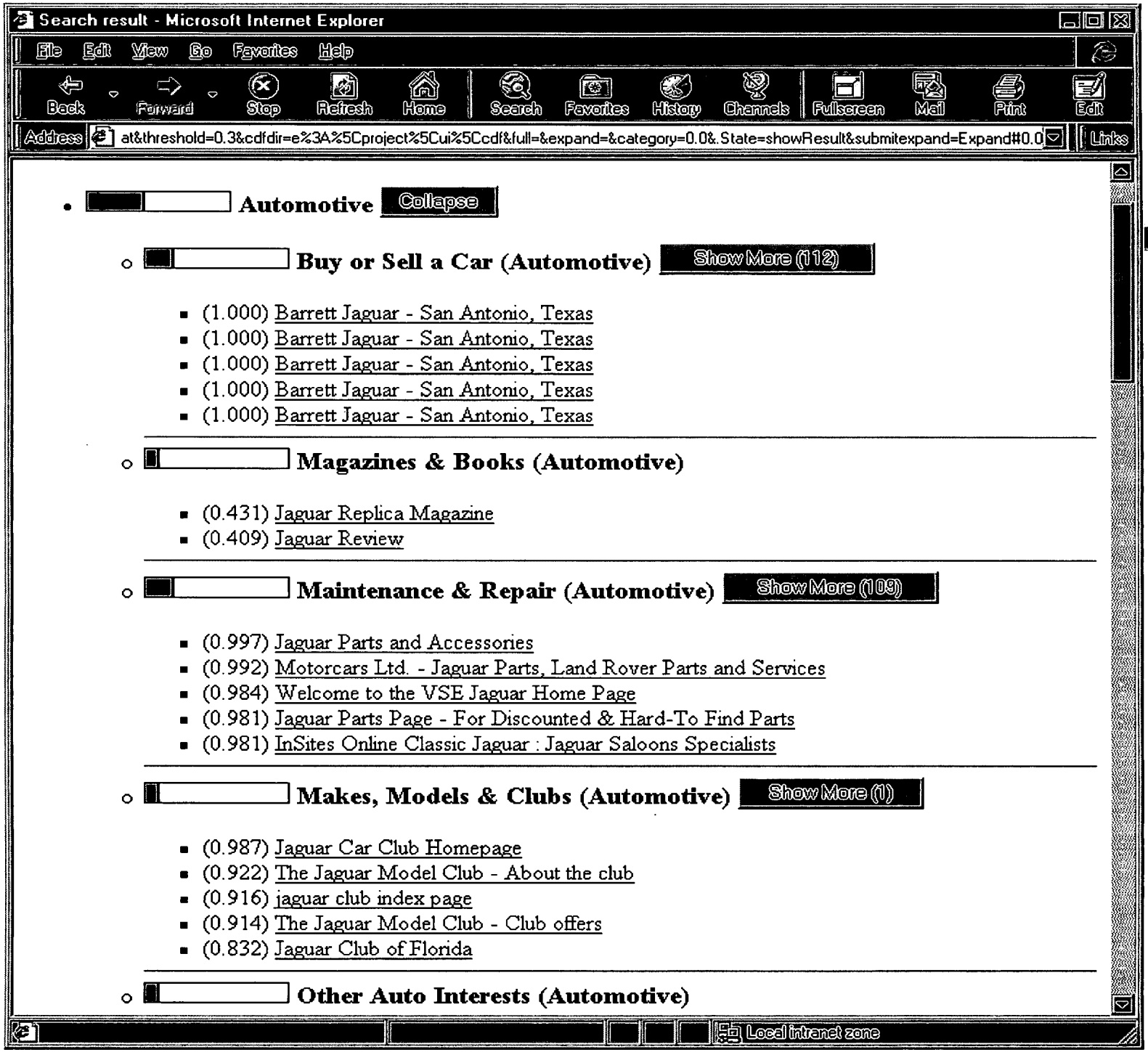
- Category Interface: 0.33 out of 30
- List Interface: 0.77 out of 30
- Significant difference, but both are small

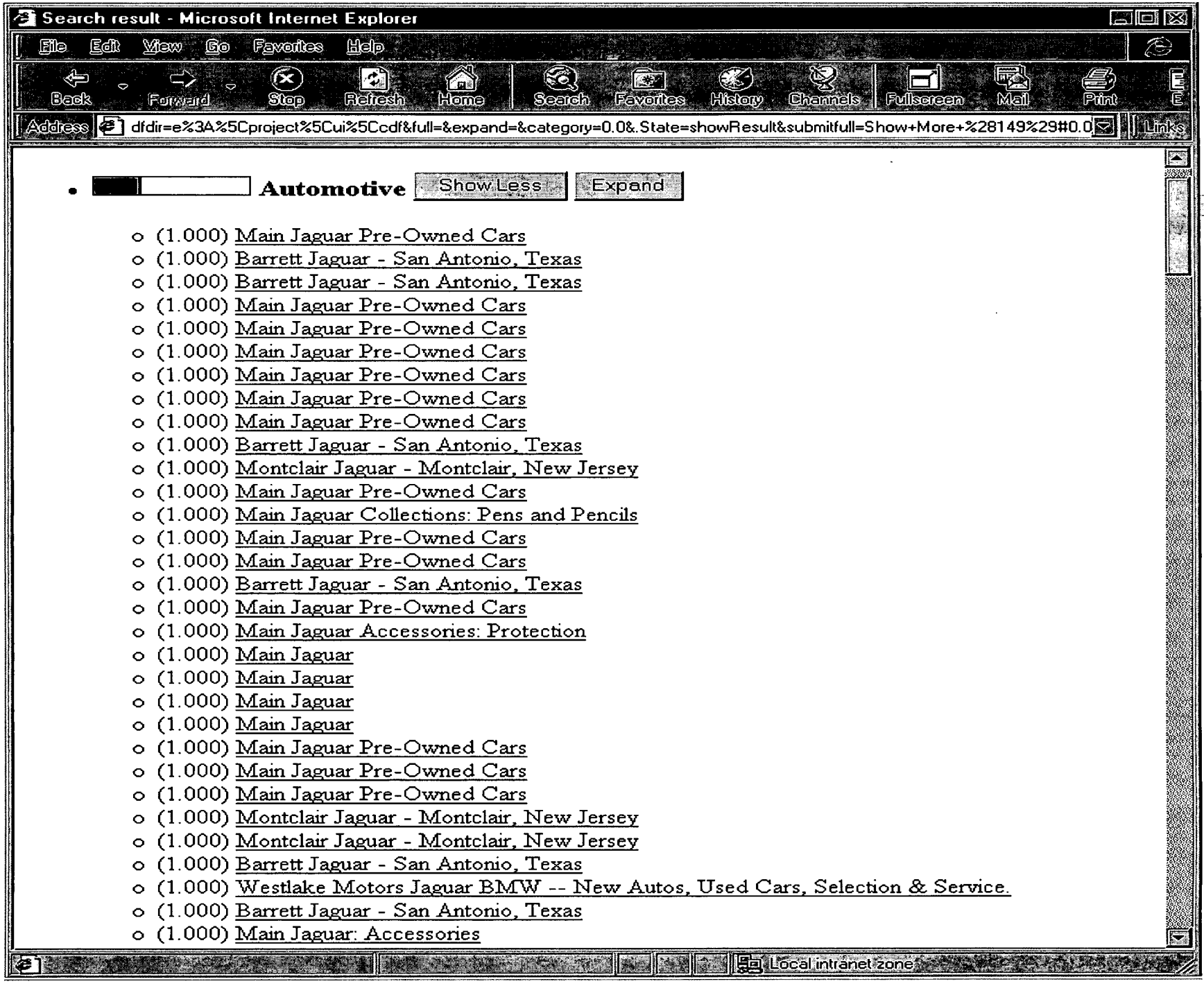
Training Classifiers

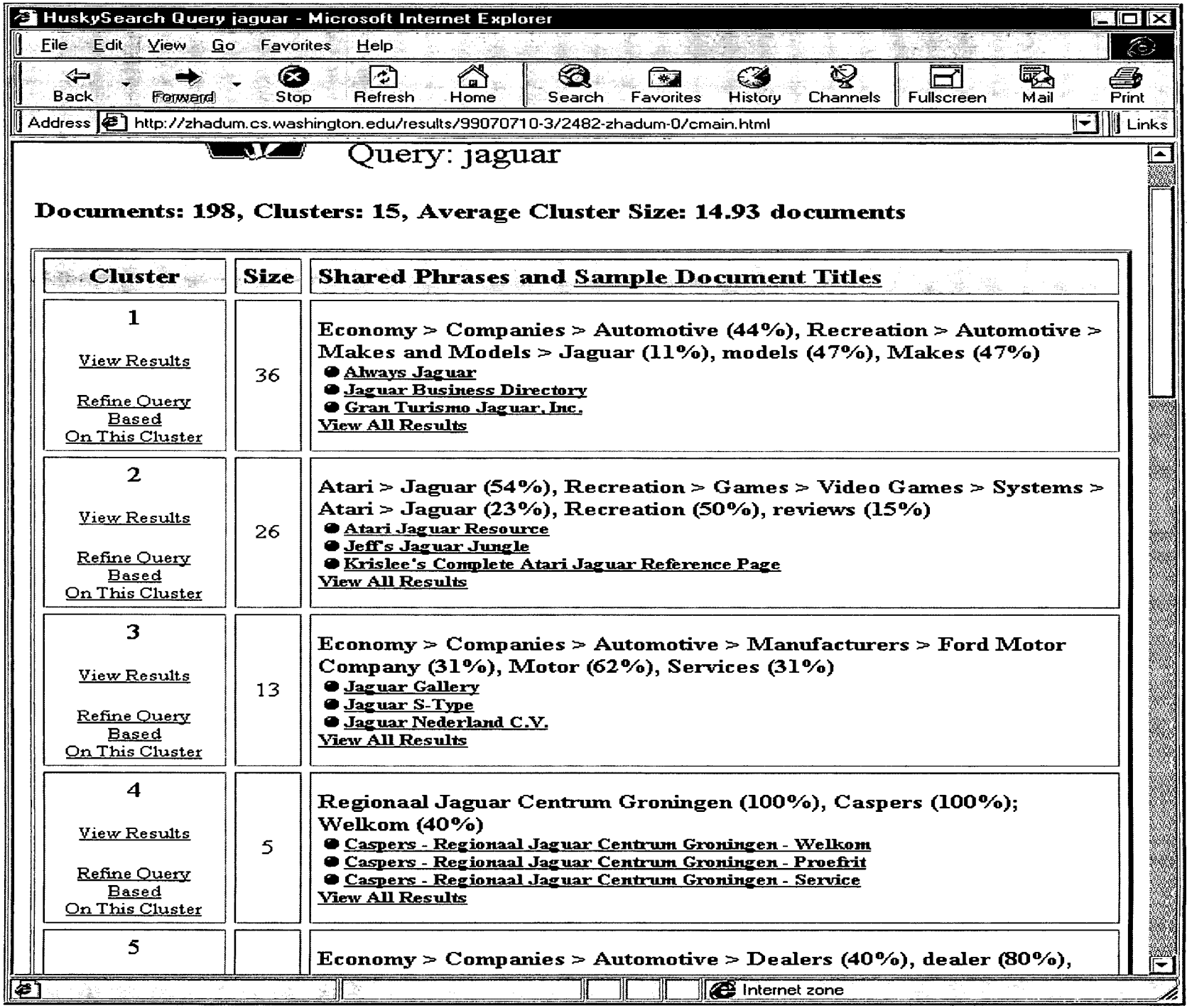
- Text pre-processing
  - download 450k Web pages
  - text = <title>, <body>, and Meta Tags (keywords, description, image text)
- Categories
  - mostly, 13 top-level categories
  - select documents w/ 250+ words
  - randomly select 10% for training/testing


Search result - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit
Address h&query=jaguar&threshold=0.3&cdfdir=e%3A%5Cproject%5Cui%5Ccdf&maxmatch=500&timeout=30&.State=submitQuery&submit=Query
Links
Query: jaguar
Retrieved 538 documents
Automotive
Show More (149)
Expand
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Main Jaguar Pre-Owned Cars
Business & Finance
Show More (13)
Expand
(1.000) Jaguar Consulting, Inc.
(0.978) Jaguar Financial Services - Online
(0.958) Always Jaguar
(0.893) Welcome to Jaguar Communications
(0.861) Jaguar Coffee - Coffee Importer and Wholesaler - Commercial - Retail
Computers & Internet
Show More (21)
Expand
(0.998) JAGUAR.NET ISP
(0.997) H-SC Jaguar Log On Procedure
(0.979) Jaguar
(0.903) Rob's Jaguar Specs
(0.881) Hasbro Releases Jaguar Publishing Rights!!!
Entertainment & Media
Show More (25)
Expand
(0.967) Jaguar Band
(0.909) NATURE: Jaguar: Year of the Cat - Print Resources
Local intranet zone


Search result - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit
Address h&query=jaguar&threshold=0.3&cdfdir=e%3A%5Cproject%5Cui%5Ccdf&maxmatch=500&timeout=30&.State=submitQuery&submit=Query
Links
Query: jaguar
Retrieved 538 documents
Automotive
Show More (149)
Expand
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Main Jaguar Pre-Owned Cars
Business & Finance
Show More (13)
Expand
(1.000) Jaguar Consulting, Inc.
(0.978) Jaguar Financial Services - Online
(0.958) Always Jaguar
(0.893) Welcome to Jaguar Communications
(0.861) Jaguar Coffee - Coffee Importer and Wholesaler - Commercial - Retail
Computers & Internet
Show More (21)
Expand
(0.998) JAGUAR.NET ISP
(0.997) H-SC Jaguar Log On Procedure
(0.979) Jaguar
(0.903) Rob's Jaguar Sp
(0.881) Hasbro Releases
http://www.jaguar.net/inet.html
Connect your business to the Internet! It's easy, when you have an experienced partner . . and the right tools Internet Access Products Turn-key offerings World Wide Web hosting Your HTML documents and graphics, our server and Internet link Web..
Entertainment & Media
Show More (25)
Expand
(0.967) Jaguar Band
(0.909) NATURE: Jaguar: Year of the Cat - Print Resources
http://www.jaguar.net/inet.html
Local intranet zone


Search result - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit
Address at&threshold=0.3&cdfdir=e%3A%5Cproject%5Cui%5Ccdf&full=&expand=&category=0.0&.State=showResult&submitexpand=Expand#0.0
Links
Automotive Collapse
Buy or Sell a Car (Automotive) Show More (112)
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Barrett Jaguar - San Antonio, Texas
Magazines & Books (Automotive)
(0.431) Jaguar Replica Magazine
(0.409) Jaguar Review
Maintenance & Repair (Automotive) Show More (109)
(0.997) Jaguar Parts and Accessories
(0.992) Motorcars Ltd. - Jaguar Parts, Land Rover Parts and Services
(0.984) Welcome to the VSE Jaguar Home Page
(0.981) Jaguar Parts Page - For Discounted & Hard-To Find Parts
(0.981) InSites Online Classic Jaguar : Jaguar Saloons Specialists
Makes, Models & Clubs (Automotive) Show More (1)
(0.987) Jaguar Car Club Homepage
(0.922) The Jaguar Model Club - About the club
(0.916) jaguar club index page
(0.914) The Jaguar Model Club - Club offers
(0.832) Jaguar Club of Florida
Other Auto Interests (Automotive)
Local intranet zone


Search result - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address dfdir=e%3A%5Cproject%5Cui%5Ccdf&full=&expand=&category=0.0&.State=showResult&submitfull=Show+More+%28149%29#0.0
Links
Automotive
Show Less
Expand
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Montclair Jaguar - Montclair, New Jersey
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Main Jaguar Collections: Pens and Pencils
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Main Jaguar Accessories: Protection
(1.000) Main Jaguar
(1.000) Main Jaguar
(1.000) Main Jaguar
(1.000) Main Jaguar
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Main Jaguar Pre-Owned Cars
(1.000) Montclair Jaguar - Montclair, New Jersey
(1.000) Montclair Jaguar - Montclair, New Jersey
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Westlake Motors Jaguar BMW -- New Autos, Used Cars, Selection & Service.
(1.000) Barrett Jaguar - San Antonio, Texas
(1.000) Main Jaguar: Accessories
Local intranet zone


HuskySearch Query jaguar - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address http://zhadum.cs.washington.edu/results/99070710-3/2482-zhadum-0/cmain.html
Links
Query: jaguar
Documents: 198, Clusters: 15, Average Cluster Size: 14.93 documents
Cluster
Size
Shared Phrases and Sample Document Titles
1
View Results
Refine Query Based On This Cluster
36
Economy > Companies > Automotive (44%), Recreation > Automotive > Makes and Models > Jaguar (11%), models (47%), Makes (47%)
Always Jaguar
Jaguar Business Directory
Gran Turismo Jaguar, Inc.
View All Results
2
View Results
Refine Query Based On This Cluster
26
Atari > Jaguar (54%), Recreation > Games > Video Games > Systems > Atari > Jaguar (23%), Recreation (50%), reviews (15%)
Atari Jaguar Resource
Jeff's Jaguar Jungle
Krislee's Complete Atari Jaguar Reference Page
View All Results
3
View Results
Refine Query Based On This Cluster
13
Economy > Companies > Automotive > Manufacturers > Ford Motor Company (31%), Motor (62%), Services (31%)
Jaguar Gallery
Jaguar S-Type
Jaguar Nederland C.V.
View All Results
4
View Results
Refine Query Based On This Cluster
5
Regionaal Jaguar Centrum Groningen (100%), Caspers (100%); Welkom (40%)
Caspers - Regionaal Jaguar Centrum Groningen - Welkom
Caspers - Regionaal Jaguar Centrum Groningen - Proefrit
Caspers - Regionaal Jaguar Centrum Groningen - Service
View All Results
5
Economy > Companies > Automotive > Dealers (40%), dealer (80%),
Internet zone

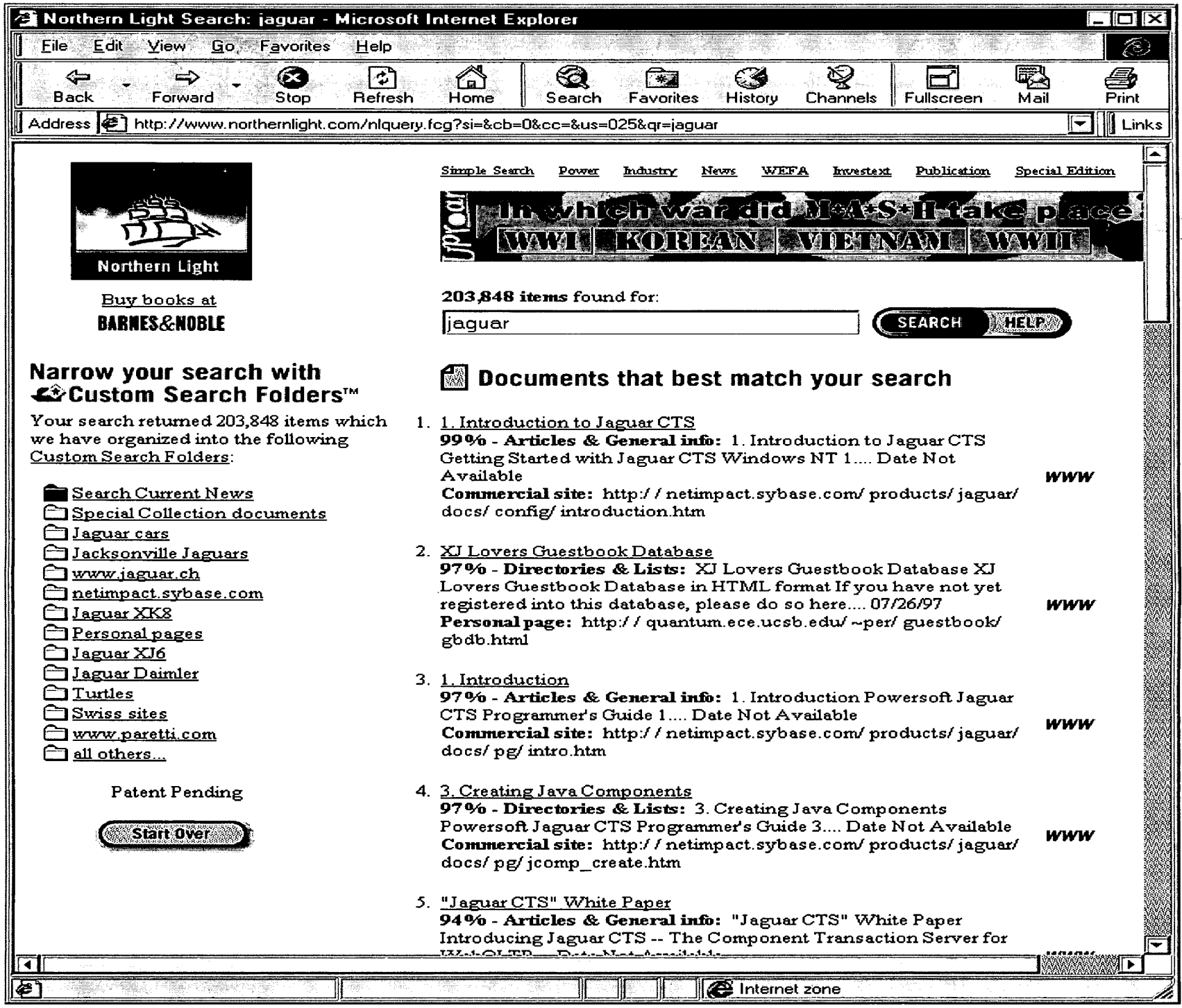

Northern Light Search: jaguar - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address http://www.northernlight.com/nlquery.fcg?si=&cb=0&cc=&us=025&qr=jaguar
Links
Northern Light
Buy books at
BARNES&NOBLE
Simple Search Power Industry News WEFA Investext Publication Special Edition
In which war did M*A*S*H take place?
WW1 KOREAN VIETNAM WWII
203,848 items found for:
jaguar
SEARCH HELP
Narrow your search with Custom Search Folders™
Your search returned 203,848 items which we have organized into the following Custom Search Folders:
Search Current News
Special Collection documents
Jaguar cars
Jacksonville Jaguars
www.jaguar.ch
netimpact.sybase.com
Jaguar XK8
Personal pages
Jaguar XJ6
Jaguar Daimler
Turtles
Swiss sites
www.paretti.com
all others...
Patent Pending
Start Over
Documents that best match your search
1. 1. Introduction to Jaguar CTS
99% - Articles & General info: 1. Introduction to Jaguar CTS Getting Started with Jaguar CTS Windows NT 1.... Date Not Available
Commercial site: http:/ / netimpact.sybase.com/ products/ jaguar/ docs/ config/ introduction.htm
WWW
2. XJ Lovers Guestbook Database
97% - Directories & Lists: XJ Lovers Guestbook Database XJ Lovers Guestbook Database in HTML format If you have not yet registered into this database, please do so here.... 07/26/97
Personal page: http:/ / quantum.ece.ucsb.edu/ ~per/ guestbook/ gbdb.html
WWW
3. 1. Introduction
97% - Articles & General info: 1. Introduction Powersoft Jaguar CTS Programmer's Guide 1.... Date Not Available
Commercial site: http:/ / netimpact.sybase.com/ products/ jaguar/ docs/ pg/ intro.htm
WWW
4. 3. Creating Java Components
97% - Directories & Lists: 3. Creating Java Components Powersoft Jaguar CTS Programmer's Guide 3.... Date Not Available
Commercial site: http:/ / netimpact.sybase.com/ products/ jaguar/ docs/ pg/ jcomp_create.htm
WWW
5. "Jaguar CTS" White Paper
94% - Articles & General info: "Jaguar CTS" White Paper Introducing Jaguar CTS -- The Component Transaction Server for
Internet zone

EXHIBIT H

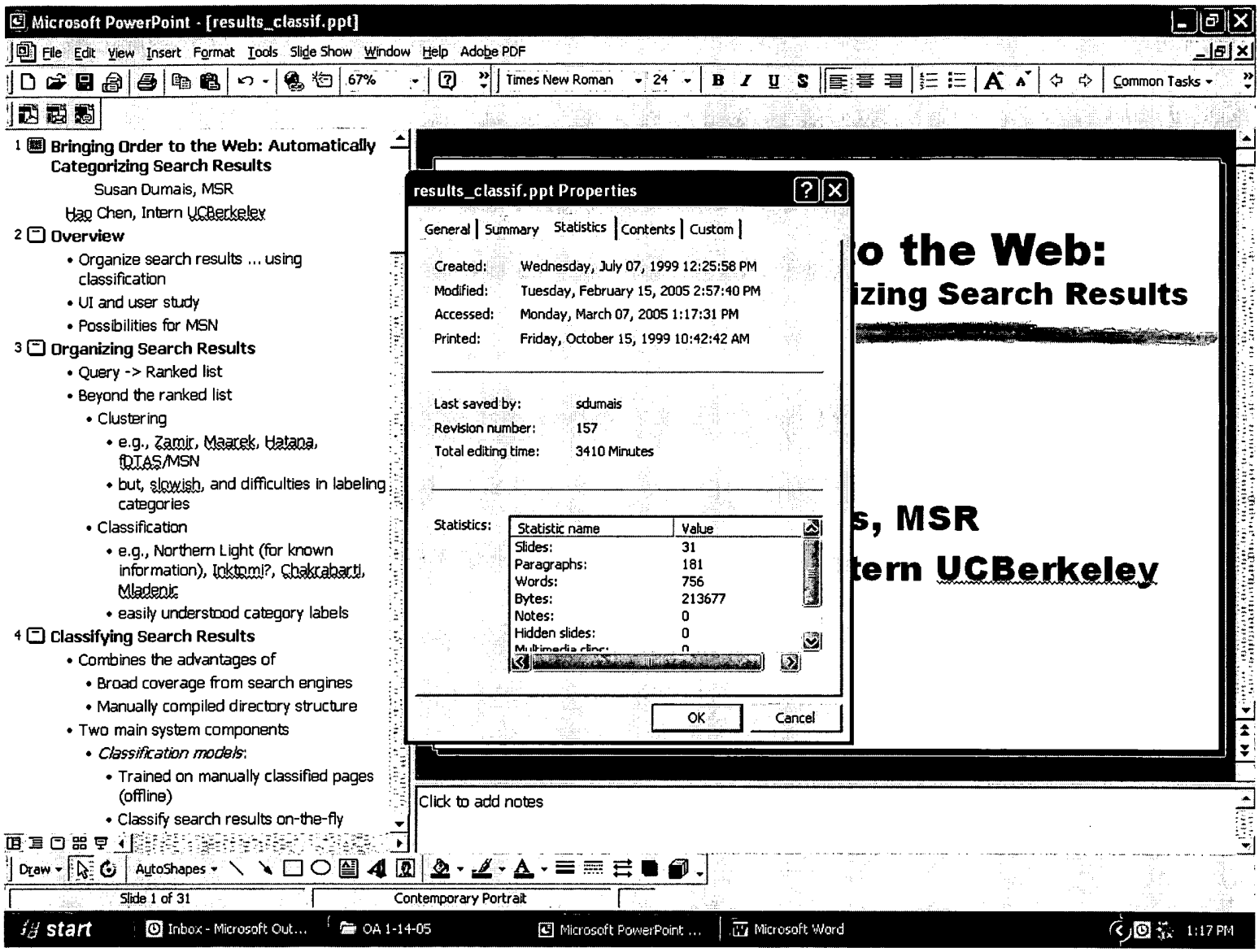

Microsoft PowerPoint - [results_classif.ppt]
File Edit View Insert Format Tools Slide Show Window Help Adobe PDF
Times New Roman
24
Common Tasks
1 Bringing Order to the Web: Automatically Categorizing Search Results
Susan Dumais, MSR
Hao Chen, Intern UCBerkeley
2 Overview
• Organize search results ... using classification
• UI and user study
• Possibilities for MSN
3 Organizing Search Results
• Query -> Ranked list
• Beyond the ranked list
• Clustering
• e.g., Zamir, Maarek, Hatana, fDTAS/MSN
• but, slowish, and difficulties in labeling categories
• Classification
• e.g., Northern Light (for known information), Inktomi?, Chakrabarti, Mladenic
• easily understood category labels
4 Classifying Search Results
• Combines the advantages of
• Broad coverage from search engines
• Manually compiled directory structure
• Two main system components
• Classification models:
• Trained on manually classified pages (offline)
• Classify search results on-the-fly
results_classif.ppt Properties
General Summary Statistics Contents Custom
Created: Wednesday, July 07, 1999 12:25:58 PM
Modified: Tuesday, February 15, 2005 2:57:40 PM
Accessed: Monday, March 07, 2005 1:17:31 PM
Printed: Friday, October 15, 1999 10:42:42 AM
Last saved by: sdumais
Revision number: 157
Total editing time: 3410 Minutes
Statistics:
Statistic name
Value
Slides: 31
Paragraphs: 181
Words: 756
Bytes: 213677
Notes: 0
Hidden slides: 0
OK
Cancel
o the Web:
izing Search Results
s, MSR
tern UCBerkeley
Click to add notes
Draw
AutoShapes
Slide 1 of 31
Contemporary Portrait
start
Inbox - Microsoft Out...
OA 1-14-05
Microsoft PowerPoint ...
Microsoft Word
1:17 PM

Exhibit I

Disclosure Meeting 10.5.00 Summary of [redacted] MS150905.1

1. Our Reference: [redacted] Microsoft Reference: MS150905.1

[redacted] Attendees: [redacted]

Eric Horvitz (Inventor)

3. Names of Inventor(s) as Initially Defined:

Eric Horvitz
Susan Dumais

5. [redacted]

5b. [redacted]

5c. [redacted]

6. [redacted]

7. Potential U.S. Critical Dates and Activities Surrounding that Determination:

Date of Conception: Before January 2000.

Implemented : January 2000.

[redacted]

6. [redacted]

[redacted]

[redacted]

8. [redacted]